

**Bauhaus Universität Weimar  
Faculty of Art and Design  
Media Art and Design**

# ***OBJECTIVE PERSPECTIVE*** ***A DIFFERENT VIEW OF INTERACTION***

**Kaël Skyhawk Hauptmann  
Weimar, September 2020**

**Vertr.-Prof. Jason M. Reizner, MFA  
Brian Larson Clark, MFA**

## **ABSTRACT**

In life people learn how many objects are used by examples, whether it be through instructions or somebody else's actions. When thinking of an object's function, the interactions are viewed through how the person or thing interacts with the object, but what if one perceived an object's application through the view of the object? *Objective Perspective: A Different View of Interaction* is a project which aims to share the function of an object via the object's perspective. This goal is achieved by making an interactive experience with the game creation system *Dreams*.

# CONTENTS

	<b>INTRODUCTION</b>	<b>4</b>
<b>demo I</b>	<b>PREDEVELOPMENT</b>	<b>8</b>
	<b>DEVELOPMENT</b>	<b>18</b>
	<b>IMPROVEMENTS</b>	<b>46</b>
	<b>PLAYTHROUGH</b>	<b>49</b>
	<b>FINAL LOGIC</b>	<b>57</b>
<b>demo II</b>	<b>PREDEVELOPMENT</b>	<b>66</b>
	<b>DEVELOPMENT</b>	<b>74</b>
	<b>IMPROVEMENTS</b>	<b>84</b>
	<b>PLAYTHROUGH</b>	<b>86</b>
	<b>FINAL LOGIC</b>	<b>92</b>
	<b>CONCLUSION</b>	<b>101</b>
	<b>OBJECTS</b>	<b>105</b>
	<b>WORKS CITED</b>	<b>109</b>

# ***INTRODUCTION***

## PURPOSE

Throughout life we are constantly learning about new objects and how they function. Some of these items are self explanatory, whereas with others we may need to be shown or taught how they function. Attempting to understand the relationship between people and objects has captivated my interest. I noticed, when we learn how to use an item, we perceive the object through how we interact with it, but what if this was not always the case?

I thought it would be interesting to apply a different point-of-view (POV) when interacting with objects. Pondering this relationship, I thought it would be intriguing to see interaction through the view of the object. I felt making a video game would be a great way of simulating this interaction and creating an entertaining experience.

## INSPIRATION

There have been games I have played which gave the player some insight into perspectives from alternative view points. One of the games that came to mind was *Battlefield 1*. The game was a first-person shooter, which took place during World War I. In it there is one small section where you play as a carrier pigeon and you have to fly to a certain destination to deliver some information to your team. Although, as the pigeon, it was in third-person perspective (3PP), I thought it was a fun little addition to the game to give the POV of a bird.

Another game that gave me inspiration was *What Remains of Edith Finch*. It was an exploration game in first-person perspective (1PP). How the game unfolds is quite interesting because you learn about the main character's family by visiting their bedrooms and playing short levels as them. In one of the stories, the character talks about becoming a cat and climbing the trees, so you become a cat and climb the trees; then you turn into a bird flying, a shark swimming, and a creature slithering.

I have been playing video games for around the last twenty-six years. The games I mentioned were the ones that came most easily to mind and consciously inspired me, but in formulating my concept I am sure that I was influenced by many others. I am sure there are films that also inspired me, like *Toy Story*.

## TECHNOLOGY

To create my project there were two bits of technology I needed, which was a Sony Playstation 4 (PS4) and the game *Dreams*. Previously, it would not be possible to create games on a video game console, it would have to be done on a computer. It is because of this that *Dreams* was created (Hanson). Mark Healey, the creative director of *Dreams*, wanted to give people the ability to easily create video games again (Hanson).

*Dreams* was officially released February 14, 2020. It is a relatively new game creation system, so they are still fine-tuning it, but the foundation is solid and people have already created amazing projects with it. The beauty of creating *Dreams* for a gaming console is that it functions on all Playstation 4s, you only need to install it and you are ready to go. This differs from traditional computers, where system requirements are to be taken into consideration when wanting to install an application.

## AUDIENCE /

## TECHNICAL LIMITATIONS

My target audience for the time being was PS4 users, who had the game creation system *Dreams*. They should range between male and females under the age of 18 through the age of 54. I decided on this range because a recent survey showed they constitute 85% of video game players (Entertainment Software Association 4). Since *Dreams* was exclusive to PS4, I would not be able to expand my audience in the future, unless they start producing it for other consoles and systems. Users also need to have their Playstations connected to the internet.

It was interesting working on this project during the time that I did. I had started it around the time the COVID-19 pandemic had hit Europe and the United States of America in March 2020. The lockdowns and social distancing resulted in much higher sales of video game hardware and games. Video game hardware sales March 2020 (\$461 million) were up 63% percent when compared March 2019 (\$282 million), and video games were up 35% from last March's \$1.19 billion to this March's \$1.6 billion (qtd. Grubb).

A large increase in hardware and game purchases meant that people would be spending more time playing video games. In March 2020, during the quarantine, gamers spent 45% more time in the United States playing video games (qtd. Shanley). They played 38% percent more in France, 29% more in England and 20% more in Germany (qtd. Shanely). Since video games have been becoming even more prevalent, it meant that my game had the potential of even more people playing it and being able to partake in the experience that I wanted to share.

## CHARACTER DEVELOPMENT

I had the purpose for my game, but I needed to start shaping the world I wanted to make. I started to brainstorm who the main character could be and what his/her relationship with objects would be. My first character idea was a person who had some kind of memory loss and would need to relearn everything. The premise would make the person to object relationship interesting, but I thought the idea was a little too extreme.

I thought about it some more and another character came to mind, a baby. This made perfect sense because at this point in his/her life he/she would naturally be trying to learn. I began researching about baby development. I wanted to find out more about when a baby begins to learn.

Babies (Birth to 2 years) start to shape basic schemas, which are cognitive representations of

past experiences and the formation of future ones (Bernstein and Nash 350). For example:

**...a sucking schema combines their experiences of sucking into images of what objects can be sucked on (bottles, fingers, pacifiers) and what kinds of sucking can be done (soft and slow, speedy and vigorous). (Bernstein and Nash 350)**

The age range I found for the baby was still too large, so I wanted to narrow it down. I looked into Jean Piaget's research on cognition in children. He talked about the first phase of mental development, which was the development of sensorimotor (Piaget 216). He broke the development of it into six stages:

- 1. birth to 6 weeks: reflexes;**
- 2. 6 weeks to 4 to 5 months: habits;**
- 3. 4 months to 9 months: coordination between vision and prehension;**
- 4. 9 months to 12 months: coordination between means and goals;**
- 5. 12 months to 18 months: discovery of new means; and**
- 6. 18 months to 24 months: insight. (Piaget 216)**

After investigating the different stages, I decided that stage 5 would fit my character's profile the best. In this stage the infant is able to learn intermediary objects accidentally as a mean to reach a distant desired object (Piaget 218).

Stage 5 also correlates to around the age when an infant learns how to walk. Infants usually start to walk on their own around twelve months and become experienced walkers around sixteen months (Adolph 363, 371). With the research I found, I thought that making my character a 16 month old girl made sense.

## PERSPECTIVES

I had determined who my character would be, but an important prevailiwperspective of the objects, it

would be in first-person. It made little sense if I was trying to give the player the perspective of an object, for it to be in third-person.

On the other hand, it seemed feasible that the playable character, herself, could be in first- or third-person. There are so many games available that use one of the two perspectives that either would feel natural to the player. I thought there would not be much of a difference on how the player would perceive the game if I made it in third-person perspective (3PP).

I based this off of my own personal experience with third-persons games where I felt connected with the characters. I did, however, think back on the first games I played which had the choice between 3PP and 1PP, like *The Elder Scrolls IV: Oblivion*. In those games I always chose to play it in first-person because it felt more natural like I had more control and was more present.

I also thought back on the *Resident Evil* franchise, which is a survival horror video game. All the major series releases were in 3PP, but the seventh game they made was in 1PP. I thought for that genre it made a lot of sense to do that because it really immersed the player into that weird, creepy, scary world, and enhanced that feeling of the actions happening to you, the player.

A study has shown that third-person perspective results in increased reaction times, higher error rates,

and higher cognitive demand because the person needs to transfer their egocentric reference frame to the avatar's position (Vogeley et al. 819). I wanted the player to feel as connected to the character as possible, so in turn, based on this evidence and my own experience, I decided that first-person perspective (1PP) would be the more reasonable choice.

When I first began this project my mission was to create one demo to showcase a different perspective. After I had completed the first demo, I made a second. The point of the second was to reuse aspects of the first but remix it into a different type of experience. I will first talk about the process of making the first demo and then the second.

demo |

# ***PREDEVELOPMENT***



# GOAL

I had defined my character, but I needed to devise a goal for my first demo. The first idea that seemed like a believable scenario was having an ice cream truck driving by the house playing ice cream truck music. The baby would hear this and want to get out of the crib, to go outside and get some ice cream. It was a nice idea, but in the end I thought it was a bit too ambitious. I needed to come up with a more reasonable goal.

I began to think about the psychology of children and how they become attached to certain objects. Toddlers can become attached to a comfort object, an object like this gives the child a sense of security and reassurance when the child's mom or dad is not present (Murkoff and Mazel 176). I thought back to when I was a child and I remembered this stuffed animal that I used to have, in the shape of a bear, and how important it was to me. I decided that retrieving a stuffed animal would be a perfect goal for the player.

The goal was in place, but I needed to figure out the scenario. I tried to imagine a few different situations in which the baby would want her stuffed animal. A feasible scenario that came to me was that the baby was being put to bed, but did not have her stuffed animal and would not be able to fall asleep without it. Her goal would be figuring out how to retrieve the stuffed animal, so she could subsequently fall asleep.

# OBJECTS

The objective was set and now I had come to the part where I needed to think about the objects with which the player would interact. I needed to find out more about people's relationships with objects, I needed to better understand affordances. Originally the term was described as:

**The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill... I mean by [affordance] something that refers to**

**both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment. (Gibson 119)**

Although this definition relates to what I am doing, I found that a later definition by Don Norman better defines an affordance in connection with my project. He described it as:

**[An] affordance refers to the relationship between a physical object and a person (or for that matter, any interacting agent, whether animal or human, or even machines and robots). [It] is a relationship between the properties of an object and the capabilities of the agent that determine just how the object could possibly be used. (Norman 11)**

I needed to keep affordances in mind in the creation of my game. I had to design it so the player would understand what each object would afford after seeing through their "eyes", because not all the objects would aid the player in completing the goal. I chose to break down the level for the first demo into three stages, each stage would have three objects with which the player would interact.

As mentioned before, the goal was that the baby was put to bed and needed her stuffed animal to sleep. Delving into that scenario some more, I thought that the stuffed animal should be sitting on a shelf that was above a baby's changing station. The baby would start in the crib and need to get to the floor, and then up the changing station to get the stuffed animal. This would give me my three stages: the crib, floor, and changing station.

I chose to first focus on the main object that would advance each stage. For the first stage, the baby needed to escape the crib and the way to do that would be by opening the crib door, so I needed a latch for that main object. On the floor, the baby would need to somehow be able to climb onto the changing station.

I figured a chair would be an ordinary enough object, but I needed to think about which type of

chair. I thought maybe a comfy sofa chair or an arm chair, but that object would be too large for a baby to move. I settled on a small stool because it is more believable that a baby could move it and it is something that could be found in a baby's room. The baby would use the stool to climb onto an open drawer on the changing table and use the open drawer to get onto the table.

Since the stuffed animal would be on the shelf above the changing station, the last object would have to be something that the baby could throw. The first object that came to my mind was a ball. It is a common object one would find in a baby's room and it is meant to be thrown.

The main objects had been selected, so I went back to each stage and thought of the other objects. The other objects in the crib seemed obvious, I needed a pillow and a blanket. It was here where I thought it would be interesting to make it so the character would not be able to jump down from the crib, unless the player had thrown the pillow on the ground. I decided later to make the blanket throwable like the pillow, but it would not help with the progression of the stage.

Coming up with the objects for the next two stages took more thought than the crib. I ended up researching images of babies' rooms to help me come up with some ideas. The two objects I settled on for the floor stage, were a trashcan and an activity table. The trashcan lid would open and shut, and the activity table would light up and play noises.

The last two items that I selected for the changing station were a baby monitor and a desk light. The light would turn on and off and the baby monitor would play static noise. All these items were typical enough for a baby's room. To help me better visualize the objects and the action of the game I created a flowchart (see figure 1).

## **OBJECT INTERACTIONS**

The objects were chosen, but how would the player interact with them? I decided on having this

connection via touch. Raccoons have very sensitive forepaws and it is believed that because of this they can differentiate objects just by touch, among other things (Zaveloff 70). Though the character would be comprehending much more than just by touch (she would be able to see through the objects' perspective), the raccoons were an inspiration.

## **CUTSCENE CONCEPTUALIZATION**

I had conceptualized most of the first demo, but I still needed to figure out the cutscenes for the opening and closing of the game, and the objects' perspectives. The first thing I did was make a layout of the room (see figure 2). I planned for the demo to begin with the baby's father tucking her into bed while she focuses on the stuffed animal and the shelf, whilst the camera slowly zooms in on it. The ending, once the baby had her stuffed animal, would be her laying back in the crib rubbing the stuffed animal, then fade to black.

I started visualizing the scenes for the objects in the crib first. I wanted to shed light on their use through their perspective. I imagined the latch scene with the camera looking up towards the ceiling from the latch's POV, with it seeing the dad come over and rotate it 180 degrees to the open position, with the camera bobbing a little bit at the end.

I wanted the player to jump on the pillow in order to get down from the crib, so I needed a visual that showed it was fluffy. The scene would again be a camera view looking at the ceiling, but in this one the dad would pick up the pillow, fluff it, and then put it back down in the crib. The purpose of a blanket is to cover someone and keep them warm, so for this scene I wanted the POV looking down at the feet of the baby, then slow move up to simulate the action of the dad pulling the blanket over the sleeping baby.

Next up were the objects on the floor. This stage would begin with the camera turning and focusing on the stuffed animal. Afterward the cutscene played, the player would be able to walk around and interact with the objects.

Figure 1. Flowchart

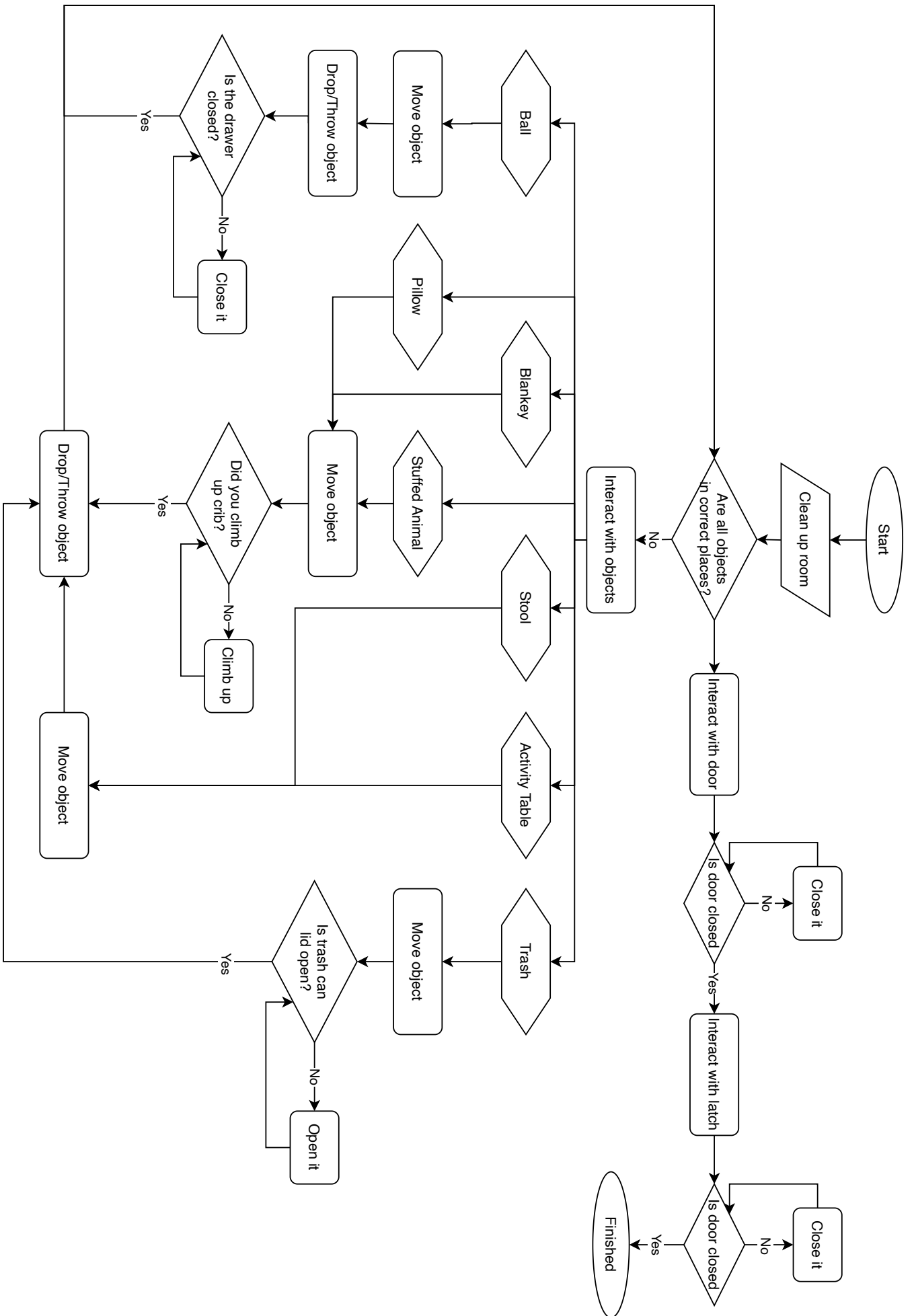
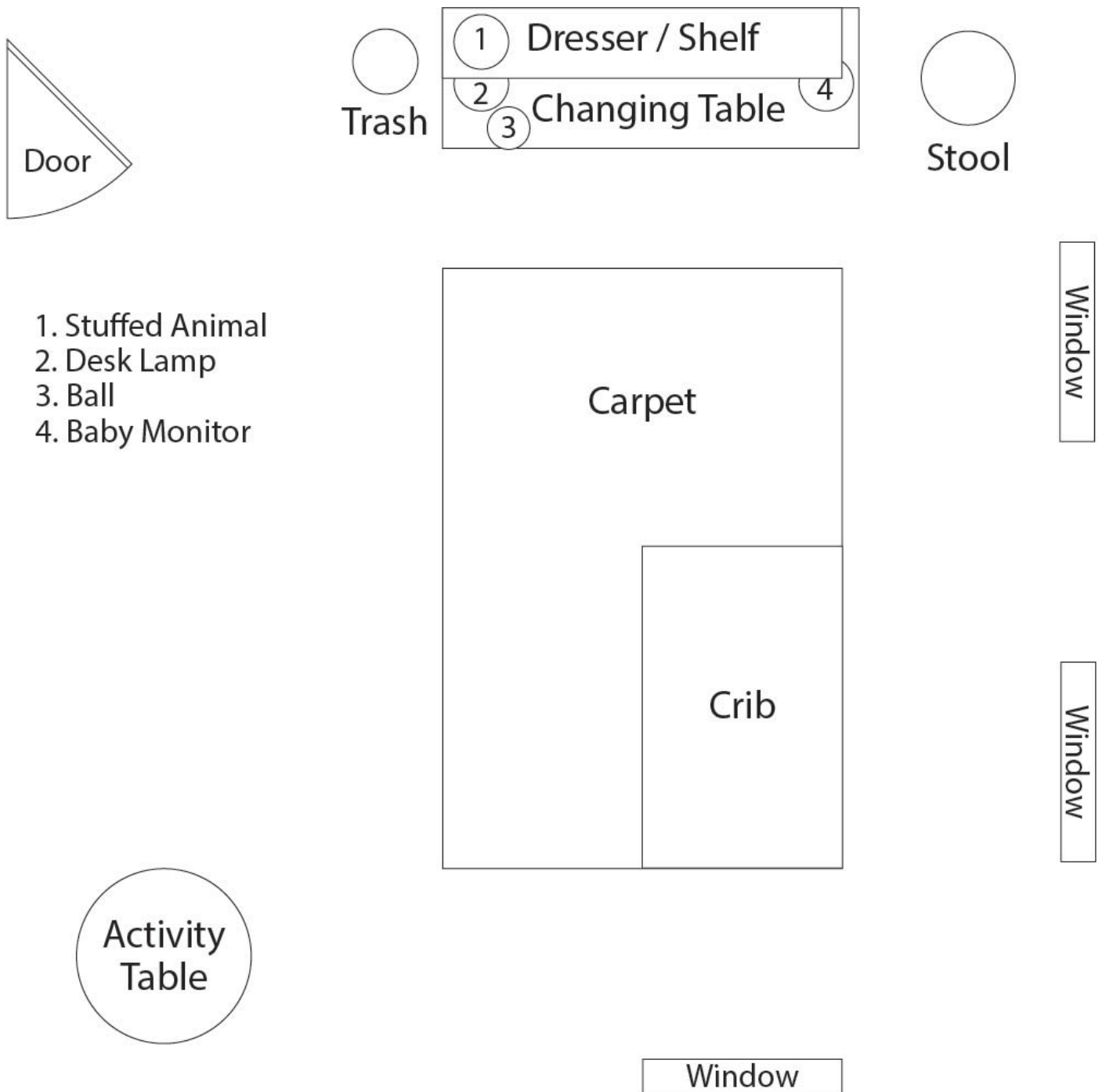


Figure 2. Room layout



People throw stuff away in trashcans, so for that cutscene the camera would be facing down inside of the can, following the lid as it opens up and the dad throws a piece of trash in it. The lid would then close and that would be the end. I already knew that for the stool I would have the perspective of somebody sitting on it. The camera would be facing upwards towards the ceiling and you would see the dad come and move the stool, then turn around and sit on it.

The last item was the activity table. For this one I would have the camera again facing upwards and you would see the dad holding the baby, looking down at the camera. The activity table would make a noise and shine a color, so you would see a faint glow of the color in the cutscene. After the noise would play and the light would shine the dad would press the button.

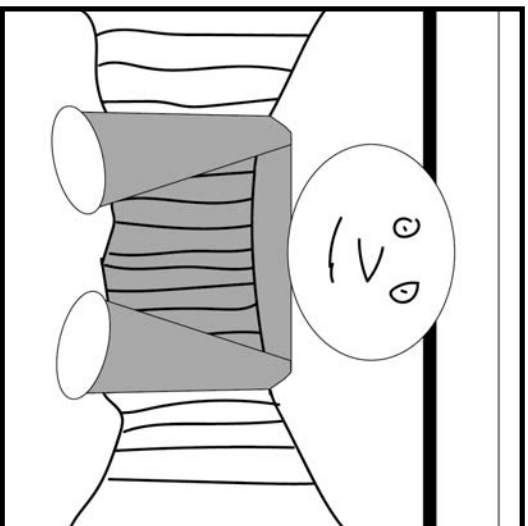
I had to think up the cutscenes for the objects on the last stage. Everyone knows lights are meant to illuminate, so for that scene I would have the camera facing from the bulb downwards towards the base. It would be dark out in the scene and you would see the dad's hand come into frame and click the light on. When the light was clicked the bulb would illuminate that area.

The purpose of the baby monitor is to predominately know if your child is crying or making a fuss. For this scene, I would have the camera facing towards the crib from the monitor. The viewer would hear the baby crying and then the father would come in and console her. Once the baby was consoled the crying would cease and the scene would be over.

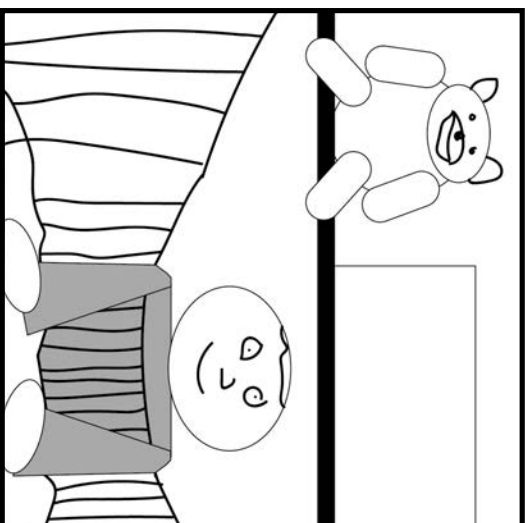
For the ball's scene, I wanted the dad to throw it. I would have the camera be in the ball facing down. The ball would be in the dad's hand and he would lift it shoulder height, then toss it up in the air a few times, then catch it. After he tossed it, he would move the ball back down to his waist.

There was not going to be an interaction cutscene for the stuffed animal, but it needed the end cutscene. The action would be that the baby would be back in the crib with it, going to sleep. The baby would be stroking the stuffed animal to exhibit her content. I made a storyboard to exemplify the key progressions of the demo's story (see figure 3).

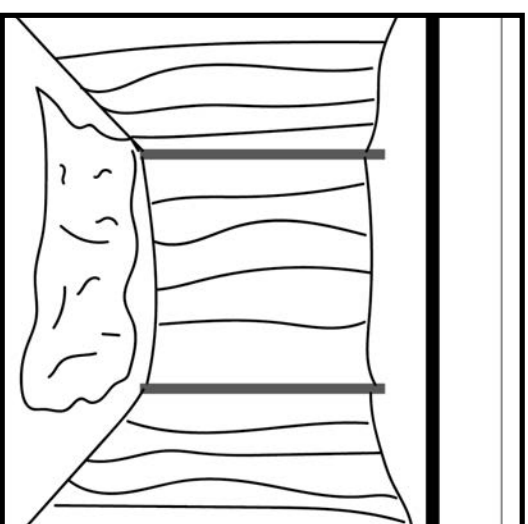
**Goal: Retrieve stuffed animal**



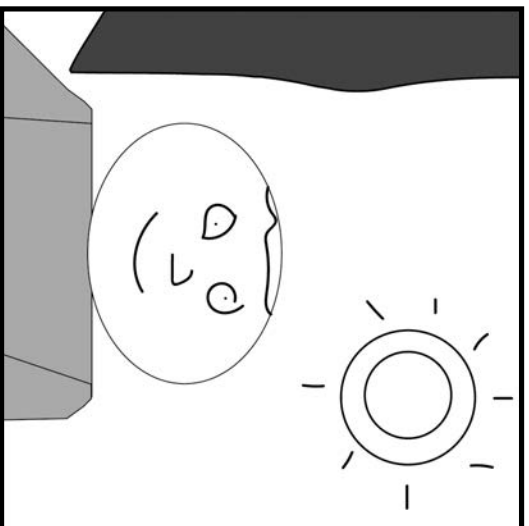
Cutscene: Father tucking baby into bed.



Cutscene: Pans over and zooms in on stuffed animal.

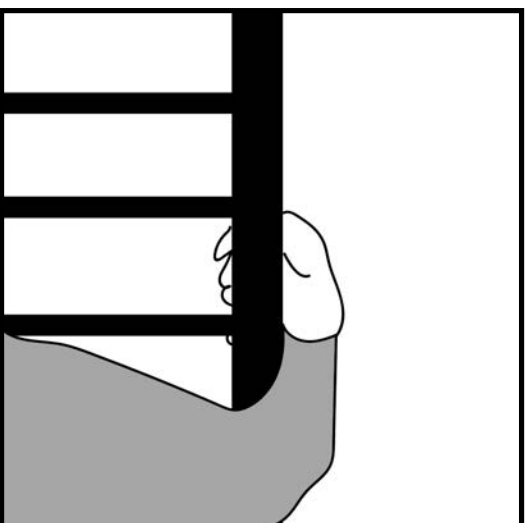


Action: Baby tries to get out of crib.

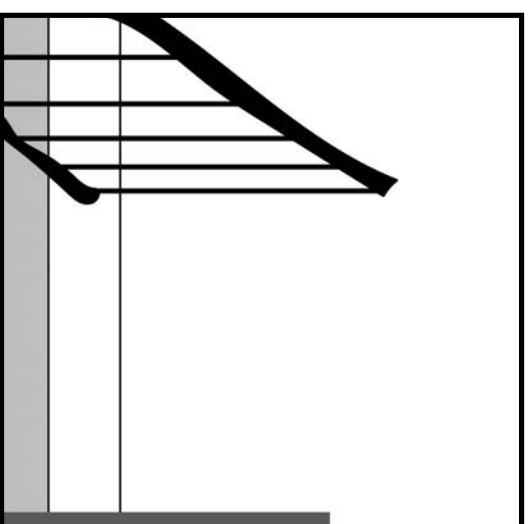


Action: Baby touches crib latch.

Cutscene: Latch POV, father closing latch.



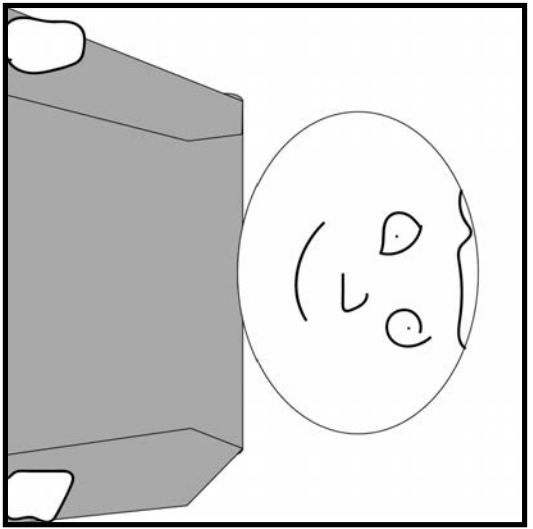
Action: Baby opens latch.



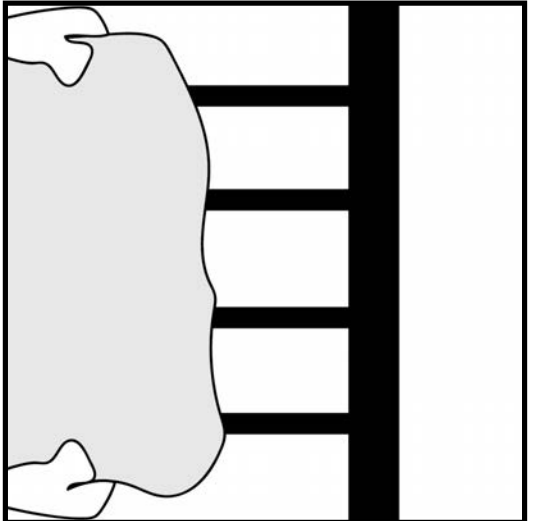
Action: Crib door swings open.

**Figure 3a. Storyboard**

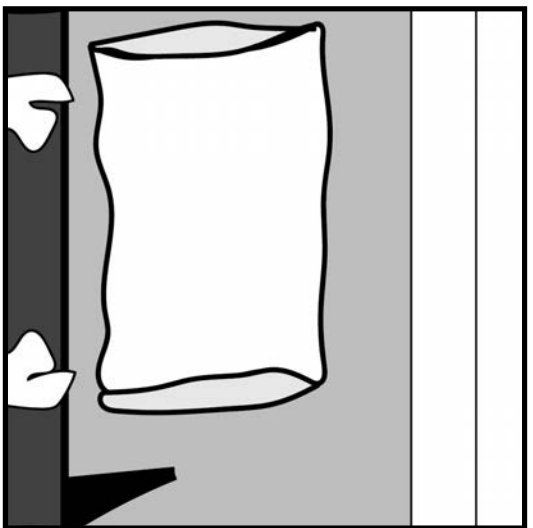
Figure 3b. Storyboard



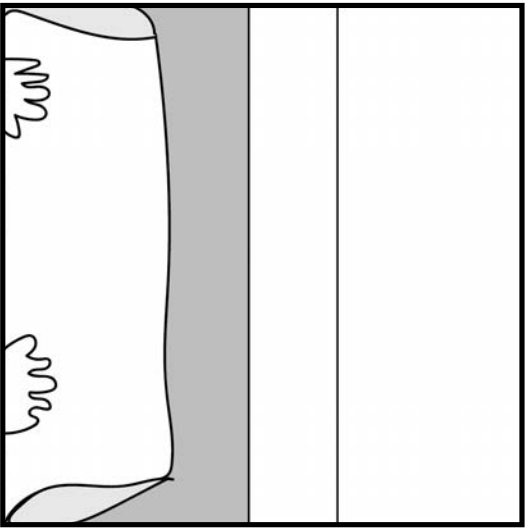
Action: Baby touches pillow.  
Cutscene: Pillow POV, father fluffing pillow.



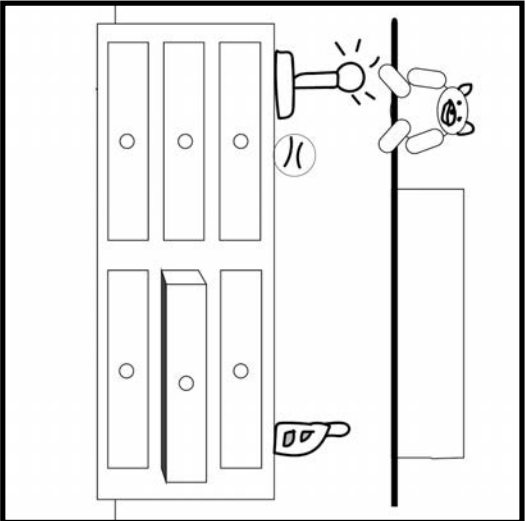
Action: Baby grabs pillow.



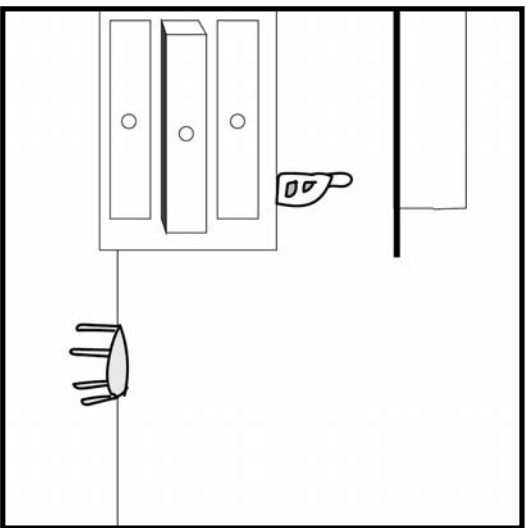
Action: Baby throws pillow down.



Action: Baby jumps from crib onto pillow.



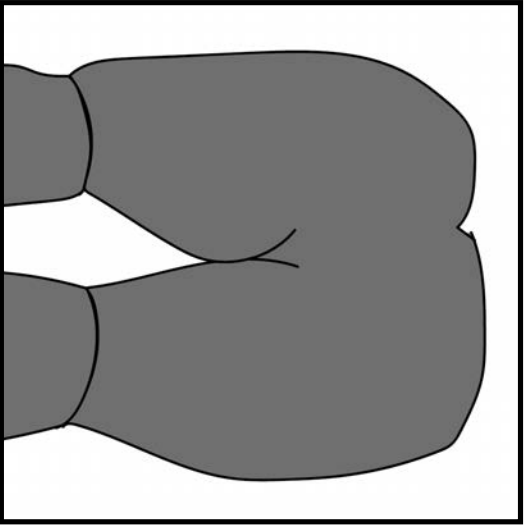
Cutscene: Camera turns right and zooms in on stuffed animal on shelf.



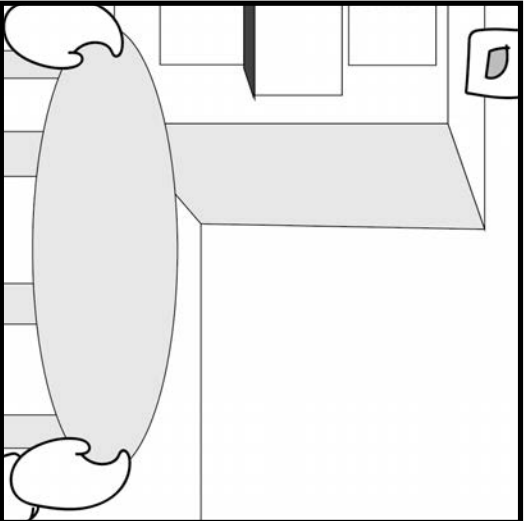
Action: Baby goes to chair.



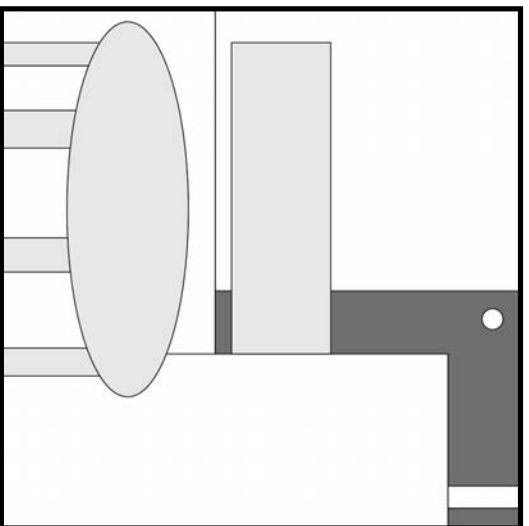
Figure 3c. Storyboard



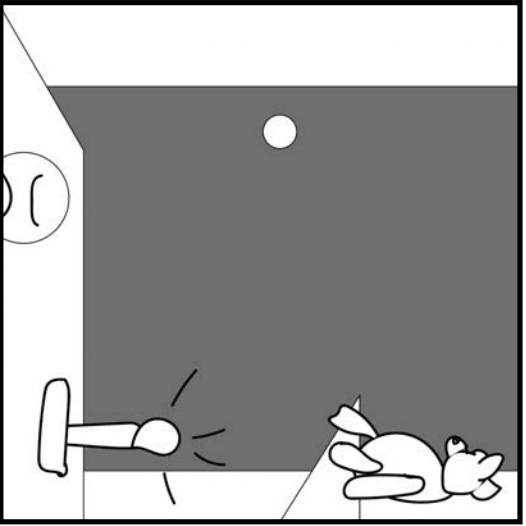
Action: Baby touches stool.  
Cutscene: Stool POV, father sitting on stool.



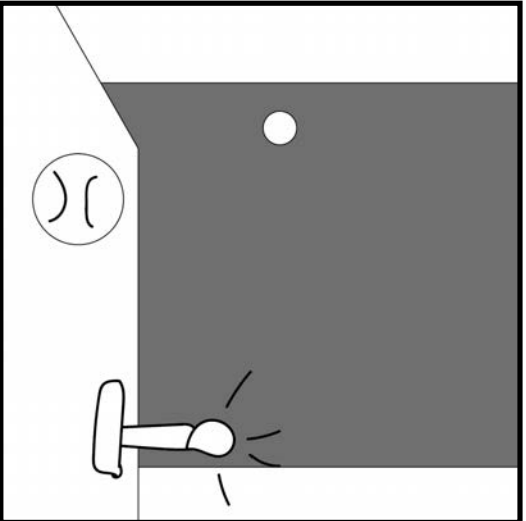
Action: Baby moves stool.



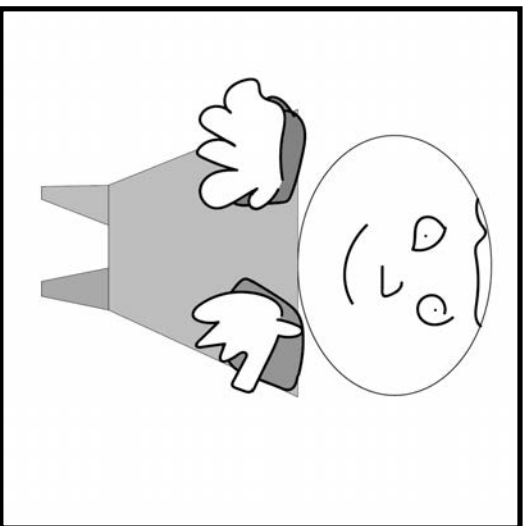
Action: Baby climbs up stool and open drawer to get on top of the drawer



Cutscene: Camera pans and zooms on stuffed animal.



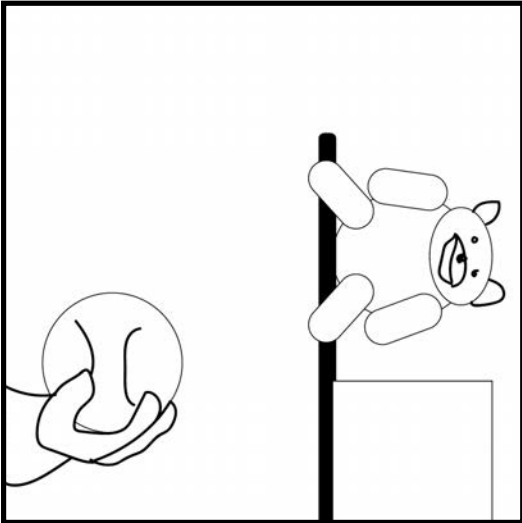
Action: Baby goes to ball.



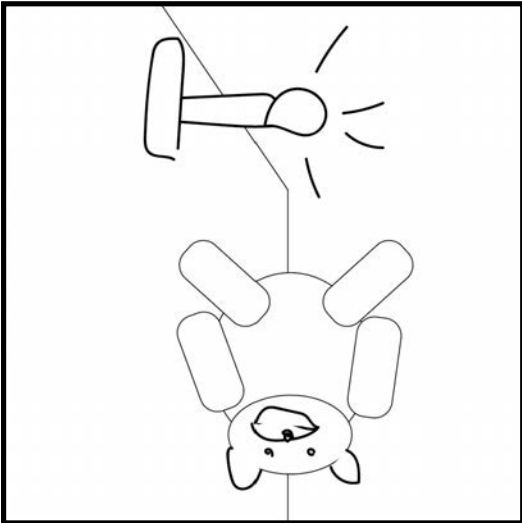
Action: Baby touches ball.  
Cutscene: Ball POV, father tossing ball to himself.



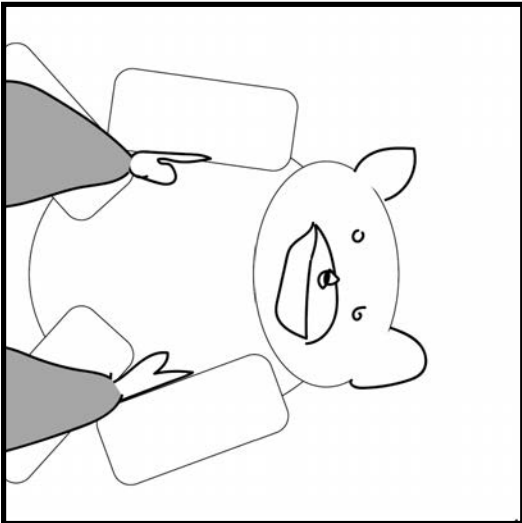
Figure 3d. Storyboard



Action: Throw ball at stuffed animal to knock down.



Cutscene: Stuffed animal falls onto dresser.



Action: Baby grabs stuffed animal.



Cutscene: Baby in bed with stuffed animal.

demo |

***DEVELOPMENT***

# TOOLS

Since *Dreams* is a new application, I would like to better explain how it works, so I can then easier explain the work that I did in it. *Dreams* can be broken down into two categories: Dream Surfing and Dream Shaping. Dream surfing is to play other people's creations and Dream Shaping is to create your own gaming experiences. The cursor in *Dreams* is an imp. The imp is used for navigation through *Dreams* and can be included as part of the games. Since I am making a video game, I will be discussing Dream Shaping.

In Dream Shaping, the creation tools can be broken down into three categories: modes, animate, and gadgets. Each category then has sub-categories. Modes is where you can create and design the objects for your game. The sub-categories are sculpt, paint, coat, style, effect, sound, and test mode.

Sculpt and paint are the main modes to create objects. The names of each already does a good job at explaining their uses. Sculpt mode is much like how an artist would sculpt an object out of clay, you use different shapes to create you object.

In paint mode, you use different patterned flecks to create objects. The flecks have a quality similar to painting something. The main difference between the two is that objects created in paint mode are never collidable in the game, whereas sculpted objects can be.

Coat, style, and effect mode are used to modify objects that you have already made with sculpt or paint. Coat mode allows you to "coat" your objects in different desired colors, hues, and finishes, as well as making the objects glow. The only problem with coat is that it applies it to the whole object.

In style mode you can change some of the visual properties of objects you have created. For example, you can change the looseness, flecks, or fleck direction of objects. Effects mode is interesting because you can add effects to objects to make them more lifelike. For example, if you have painted a

stream, you can add a flow effect, which makes it look as if the water is flowing in whichever direction you have chosen.

Finally, there is sound mode and test mode. You can use sound mode to create your own music, sounds, and dialogue. You can also import different audio that has been uploaded on the *Dreams* network by other users.

*Dreams* has its own music creation studio. It reminds me of a more visual, simplified version of a software program like *Logic Pro* or *GarageBand*. Test mode lets you play the game, while still displaying all the development-side elements.

The next category I am going to delve into is animate. It has three main components: action recorder, keyframe, and record possession. Although all three deal with actions of elements in the game, there are subtle differences between them. The action recorder is used to record movements or actions of elements, which you do with the controller. Record possession is similar to this, but in record possession you can possess a puppet/character, so you can move it with the controller like you were playing it in the game.

Keyframe mainly differs from the other components because it does not continuously record. An element can have a keyframe for the starting position and the finishing position. By default the element will be in the position of the first keyframe and then instantaneously move to the position of the next keyframe. You can blend the positions, but the way keyframes are, they will move the element in the most linear fashion from one position to another. All of the components can be organized and sequenced in a timeline.

Gadgets are the last category. Gadgets are the technical pieces used to construct your game. They are divided into a few sub-categories. There are sensors & inputs, logic & processing, movers & output, gameplay gear, cameras & lighting, connectors, and sound. I am not going to explain each gadget

inside of the sub-categories, but I will give a general explanation of the different sub-categories.

Sensors & input are gadgets dealing with when you want something to happen because of something else. What these gadgets do is generate signals, which are carrying values. They usually signal the beginning of an event chain.

Logic & processing is pretty self-explanatory. Here you create your rules and conditions for the different elements in the game. They typically are in the middle of the event chain. Movers & output tend to be at the end of an event chain. They use the processed signals that were made by the sensors and perform the desired function.

Next is gameplay gear. These gadgets deal with typical gameplay functions, like checkpoints, scoring, and prize bubbles. Blank puppets and the puppet interface gadget can be found here. Puppets are blank characters that have a bit of predefined properties in them that you can use for the game.

Cameras & lighting consists of gadgets, which produce cinematic effects. You can arrange the different camera angles you want for cutscenes and gameplay. You can set up the ambience with lighting, weather, grades and effects.

After cameras & lighting, there are connectors. Connectors are used to join different objects together. Depending on what type of connectors you use and where you put them determines the type of motion for the objects joined.

The last gadget is sound. In sound you can place sounds in different places in your game. You can add reverbs, delays, and other effects to your sounds. Unlike sound mode, sound only deals with gadgets to modify sounds in your game, sound mode has many more features. As seen with sound and sound mode there are bits of repeating tools throughout the interface.

I have discussed most of the tools, the only parts left are the assembly tools, guides, show/hide, and search. The assembly tools offer typical functions

you would expect when dealing with elements. You can move, stretch, clone, delete, tweak, hide, etc. Guides allows you to turn on grids, auto-guides, grid snap, precision move, stay upright, and change the lighting, while designing the game (edit mode). Show/hide does not need much explanation. You can show or hide various elements and gadgets in your game in edit mode.

Search is a nice feature they have in *Dreams*. *Dreams* has already created myriad of assets that users can use, but they have built up a solid community, who also upload assets, which are usable and searchable. Although this is a very convenient feature, I elected to create all my assets myself. Now that you better understand how *Dreams* works, let us move on to the development stage.

## **FIRST STAGE**

I had already segmented the demo into three stages, so I would develop it stage by stage. In the beginning development I would not be focusing on the detail in the objects, but rather the figuring out the logic and being able to run the game. The first stage was escaping the crib, so the first step was to construct the assets needed. This meant the crib, pillow, blanket, and character. When you begin a project in *Dreams*, there is already a floor generated, so I could begin working on the other items.

I made the crib in sculpt mode. I edited the cube and cylinder shape to make the various sections that constituted the crib. Since measurements and orientation were important I used the a grid and snapped the objects to it. Mirror and clone helped expedite the process.

The first real obstacle I encountered was creating the door for the crib. When you create in sculpt mode, all the elements are contained in one group automatically. I wanted to add connectors to the door, so it could swing open, but it would not let me.

I then remembered that objects cannot be in the same group if you want to connect them because one needs to be the parent and the other the child.

I took the door out of the group that the rest of the crib was in and then I was able to connect them. I had to mess around a little to find the correct connector to use.

I ended up using a bolt because they are designed to rotate connected objects around an axis. I used two bolts, one close to the top and one close to the bottom, so the door would swing properly. I added a 180 degree limit to them.

With the door properly in place, the next step was adding the latch, which would be opened by the baby. I used the cube shape to add the frame to hold the latch within the group that contained the rest of the crib, and then I used the cube shape to create the latch separately. Since I already had done it with the door, I connected the latch to the door with a bolt connector and made sure the latch could rotate from closed 180 degrees to open. I would still need to add logic to the latch for the open function, but this was done later. I also used sculpt mode to make a mattress. See figure 4 for baby crib example.



Figure 4. Baby crib

## FIRST PERSON PUPPET

The next asset I built was the baby. In *Dreams* there are the premade puppets I mentioned earlier, which have logic predefined, like body movement, body structure, puppet behavior, physical properties, etc. I selected a blank puppet. The first part I changed on the puppet was its dimensions. I wanted it to be a little girl, so I compacted the size by using the stretch tool and adjusting the arms, legs, torso, and head accordingly (see figure 5).



Figure 5. Compacted puppet

By default, the point-of-view for puppets is third-person, so I needed to change it to first-person. You also do not automatically start off as the puppet, you still have your imp and you need to possess the puppet. This needed to be changed as well.

To begin this process I turned on the grid snap and made two cubes in sculpt mode, one on top of the other with a little bit of a gap in between. I then added a bolt connector to attach the bottom cube to the top one. I adjusted the connector, so that the top cube would rotate vertically 360 degrees.

I snapped a microchip to the top cube and one to the bottom. A microchip is a gadget which helps you organize your gadgets and wires, similar to placing all the files on your desktop into a folder. Wires are how you connect gadgets.

The first gadget I put inside the top microchip was a camera gadget. I adjusted the camera view, so it was in the middle of the cube just poking out. Since the camera is in the microchip that is snapped onto the top cube, whenever the cube moves, the camera will move with it.

The camera was going to be the view for the player. I also turned the visibility off and made them non-collidable. I chose those options, so the cubes would not be seen in the game and there would not be any clipping issues.



The next thing I did was entering the puppet's microchip and adding a "Player" tag. Tags are used to designate specific locations. I moved the location to the puppet's lower chest area. On the bottom chip I added a teleporter and had it target the "Player" tag.

The tag was so the cubes would always remain at the same position of the puppet when it moved. I also placed a gyroscope in the bottom chip and boosted the strength and overall damping to 100% and raised the speed up to 280.0 degrees/second to aid in keeping the cubes upright.

The blank puppet already has a controller sensor gadget in its microchip with some predefined settings. I needed to change it, so that when you play the game, you automatically start off as the character. I was able to achieve this by going into the controller sensor's options, under important properties & I/O, and turning on force possession. I also deleted the wire that made the circle button de-possess since I wanted the user to always be the designated character.

The left stick for the controller was already defined for walking around, but I still needed to configure the right stick, so it would look around. I began by adding a wireless transmitter gadget called, "PlayerRightStick", in the puppet's microchip and connecting the right stick from the controller sensor with a wire. I would use this signal to configure the looking on the y-axis and the x-axis.

The logic for the y-axis I stored in the top cube's microchip. I placed a wireless receiver in there and set it to pick up "PlayerRightStick" and set the zone for the entire scene. In the microchip I also placed a timeline, which I set to ten seconds. I created three keyframes: the first was the lowest position that the camera would face looking down, the second was facing directly forward, and the third was highest position facing up.

I blended the keyframes linearly and made sure smoothing was on (see figure 6). This is to mimic the first-person view of somebody looking down as far as he/she can and then slowly looking up as far as he/she can. It is important to have the keyframe of

looking straight forward right in the middle of the timeline at five seconds.



**Figure 6a. Upward camera keyframe**



**Figure 6b. Downward camera keyframe**

Next I snapped a variable gadget in my chip and labeled it, "CameraSensitivityY". I set the minimum value in it at -1.0 and the maximum at 1.0, which are the max. and min. values that the y-axis of the right stick can output. The initial value I set to -.5. I made it negative because if it was a positive integer up/down would be inverted, which was not what I wanted.

Now going back to the signal from "PlayerRightStick", I had it go into a splitter and had the splitter only send out the signal from the Y into a calculator. The calculator is set to multiply, which will multiply how much we change the speed of the timeline. I created a variable modifier and had

it get the “CameraSensitivityY” value and send it as the second operand to the calculator.

After that I created a value slider. The minimal value for it was 0 and the maximal was 1, so I set the default to 0.50. This is so the playhead will start in the middle of the timeline where the camera is face straight forward. I wired the slider into the playhead of the timeline.

I snapped another calculator, set to add, into the microchip and had the signal from the multiplier output into it. I then had the output of the value slider go into the add and have the add calculator output into the value of the slider. I did this, so I could add and subtract from the value slider, but have it remain at a certain value if the right stick was not being used.

I had my logic in place, so it was time to test to see if the up/down look of the right stick worked properly. I switched over to view mode and tried it. The stick did perform the proper function, but it moved too fast for my liking, so I went back and changed my initial value for “CameraSensitivityY” to -.10. I tried this and although it was better, it still moved too quickly. In order to fix this I needed to create some logic to get a finer value.

I created a new slider, called “YSensitivity”, and set the initial value to -.50, the minimum value at -10.0 and the maximum at 10.0. I then connected to a calculator, which I set to divide “YSensitivity” by 10. I output that to the operational value of a variable modifier, which would set “CameraSensitivityY” to that value. I added another calculator and set it to equals zero and connected the divide calculator’s result to its operand 1.

I connected the output to a NOT gate and output that to the power of the variable modifier. That means that when the “YSensitivity” is 0, the variable modifier would be off. I tested up/down again and it worked a lot better, but it was every so slightly too quick, so I changed “YSensitivity” to -.40, which seemed to work out pretty good (see figure 7).

The y-axis look was setup, the next step for me was setting up the x-axis. Much of the setup was similar to which I did in the microchip for the y-axis. I opened up the microchip in the bottom cube; it already contained the teleporter and gyroscope I mentioned before. I added a variable called, “CameraSensitivityX”, set it to -.50, set up a wireless receiver that would pick up “PlayerRightStick” and inputted it into a splitter. This time I had the splitter output the left/right signal into a multiplier calculator. The other value input into the calculator was “CameraSensitivityX”. I sent the product of these numbers to a rotator.

A rotator is used to spin objects around on a single axis. I set the rotation speed to 360 degrees/second and pushed the strength and damping up to 100 percent. I also dragged the rotator up, so it would spin on the x-axis.

I tested it and saw that the cubes would not follow the puppet when I rotated. I fixed this by putting a tag in the bottom cube, called “PlayerForward”, and setting its location right in front of the camera. I went to the puppet’s behavior, turned off follow and look at, and set turn towards to “PlayerForward”. This fixed the problem, but the puppet’s limbs would flail in an undesirable way.

I went back into the puppet’s properties to change the puppet’s upper body and lower body settings. For upper body I turned down motion sensor movement, lean lag, sway, lumberingness, stiffness, arm flail, and springiness to zero. For lower body I changed lean into strength to zero.

By changing these settings the puppet moved more naturally, but the x-sensitivity was too high. I copied the slider, two calculators, variable modifier, and NOT gate from the top cube into the bottom. I changed the name of the slider to “XSensitivity” and had the modify variable modify “CameraSensitivityX”. I tested the x-sensitivity and it moved at a good speed (see figure 8).



Figure 7. Camera rig y-axis logic



Figure 8. Camera rig x-axis logic

## THE LATCH

As I stated previously, before the player interacts with the object, I wanted the player to see the function of the object via the perspective of the object. That meant the microchips I created for the objects would contain the logic for the cutscenes and for interacting with them afterwards. I had my first-person puppet and the crib with a latch. My next objective was to create the logic and animation for the crib latch.

I began with a trigger zone; I set it to detect “Player” and set the zone shape as a rectangular prism in front of the latch. I connected it to a text displayer. In its properties, I changed it to display a green triangle, I turn off the text box and border, toggled it to be in-scene, facing the camera, and to always be on top. I also changed the size and location for the text, so it was right on the inside of the crib door next to the corner with the latch. Apart from the actual text and color, I used these same settings for all my objects that required the player to press something.

I added a controller sensor as well since I wanted the player to press a button to perform the task. I had the wire from the square button go into an AND gate and I put a wire from the trigger zone to the AND gate. An AND gate will not send a signal through unless all the inputs are sending signals. I set a counter with a target value of one. Once the AND

gate sent the signal to the counter, it would increase the value to one and the counter would send a signal.

I created a keyframe of the latch at the open position. Since I already had a bolt on the latch, when it performs the keyframe it will rotate the latch around the bolt. In the keyframe’s settings, I set the slow power up and slow power down to five seconds, so that the latch would open slower. Another important thing I did was turn keep changes on, so that after the keyframe played through the latch would remain open in the scene.

Initially, I had the output of the counter connect to the keyframe. This worked when I tested it, but I realized later that the counter was unnecessary. I deleted the counter and connected the AND gate directly to the keyframe.

I noticed that when the latch opened the gate still stayed closed. It could be opened if the puppet walked into it, but that was not what I wanted. I wanted it to swing open when the latch was opened. To achieve this I inserted a timeline, within which I put a keyframe in the beginning with the crib door closed, and a keyframe at the end with the door open.

I turned keep changes on and selected a linear blend between the two keyframes. In the timeline’s properties, I selected sustain under playback mode, so the timeline would only be played once and the changes would remain. I separated the keyframes by ten seconds, but



the door would swing open too quickly, so I tinkered with playback speed in the timeline and found that at 50.00% it opened slowly enough.

I outputted the AND gate to the timeline as well, but a problem I realized was when I opened the latch the gate would open before the latch was opened all the way. To fix this I added a timer between the AND gate and timeline. I set the timer to one second, but when I connected the timer finished (pulse) output to the timeline, nothing would happen. I put a counter in between them and then the timeline played properly with the delay.

I had the logic in place to open the latch, but the next step was adding in the cutscene before the latch was opened showing the latch's POV. I started by disconnecting the wires from the AND gate to the keyframe and timeline. I would reconnect them later.

I made two cubes in sculpt mode, placed them where the latch was, and connected them with a bolt. I modified the angle and rotation, so it would mimic opening like the latch. I made a 180 degree limit and set the springiness of the bolt to 30 percent. I snapped a microchip onto the cube that would be rotating around the other one.

Inside the chip I put a node, which is basically a gadget that passes whatever signal is sent through it and is used for tidying up the wiring. I connected the AND gate from the trigger and controller sensor to it and connected it to a new timeline. In the timeline I added a camera gadget. I adjusted it, so the camera was facing up to show the POV of the latch when it was closed. I also changed the transition type to cut.

I created a keyframe in the timeline of the camera face up, then i created a keyframe where the cube had rotated to the other side mimicking the opening of the latch. Since the microchip was snapped onto the cube the camera followed the cubes 180 degree rotation. I added a few keyframes of the cube bobbing up and down at then end and I blended them together linearly.

In this cutscene I wanted the father to move the latch, so I added a blank puppet in the middle of the scene. I did not want the puppet to be seen during the gameplay, so I went to his properties and turned visibility and collision off. I turned preview invisibility off in the scene, so that I could still see the puppet.

I made a keyframe where I made the dad visible and the baby invisible. I extended the keyframe for the entire timeline. I made the baby invisible because I do not want her to interfere with any of the cutscenes.



Figure 9a. Puppet's movements in timeline

I used the possession recorder to possess the father and have him walk towards the crib. Below on the timeline, I added a few more keyframes to move the dad, so it looked like he was turning the latch. I made the blend linear and adjusted their positions in the timeline to sync with the turning of the latch (see figure 9).

I had it connect, so when the player went to the door and pressed triangle, it would play the cutscene, but I needed to figure out how to connect pressing triangle to open the latch afterwards. I had the timeline go into a node that was in the other microchip. I connected this node to a counter, which I set to one. I then connected it to an AND gate; I had the first AND gate for the trigger zone and controller sensor connect to it too. This meant that the second AND gate would not work until after the cutscene played. I connected it to the latch opening keyframe and the other logic for the gate to open.

The logic worked and the latch first played the cutscene and then the character could open it, but after the gate was open, it would still show the triangle display when the player was in the trigger zone and during the cutscene. I needed to change that. I accomplished this by using a selector. A selector has different slots for input and output. An important rule to note is that A output will always send out a signal even it is not receiving a signal.



Figure 9b. View from latch

I connected output A to power the text display; I also had the counter, which sent a signal to the scene, send a signal to output B in the selector. This would make it so when the cutscene plays the selector is on B and would not output power to the text display. Since I wanted, though, for the triangle to show back up in the scene after the cutscene I had the cutscene timeline send a signal to the selector to move to previous output. With that the text display would be on again; I then used the second AND gate, which connected to the logic to open the gate, and had it send a signal to the selector to move to next output, which would again turn off the text display (see figure 10 for logic).



Figure 10. Latch logic

# THE PILLOW

The next task I needed to do was create a pillow that the player could pick up and throw. I went into sculpt mode and adjusted the cubes dimensions, so that it was a rectangular prism that resembled a pillow. I placed the pillow on top of the mattress in the crib. I snapped a microchip onto the pillow and began with a similar approach as the latch.

I used a trigger zone, set the zone to a box, and had the zone envelope the pillow with some extra space on all sides. I copied the text displayer gadget from the latch and placed it in this chip. I changed the text to display square, I changed the color to pink, and I moved it, so it would display above the pillow. I output the trigger zone to the text displayer and to an AND gate. I added a control sensor and sent the square output to the AND gate which then outputs into a counter with a target value of one.

I had the counter output to a teleporter. The reason I did not have the AND gate go directly into the teleporter is because I needed a constant signal to keep the teleporter on, without the counter, the player would have to hold square to keep the object in the character's hand. I turned on match target position and match target orientation, and I also adjusted the teleporter's reference point to the center of the pillow.

Since the teleporter needs a target which to teleport, I scoped into the baby puppet and place a tag called, "Chest", on her abdomen. I placed the reference point of this tag a little in front of the chest. This worked to teleport the pillow, but when it teleported to my character it did not have the proper orientation, so I needed to fiddle with the reference point of the teleporter and change the axes a little.

I had made the logic to hold the pillow, so I needed to make the logic to throw it. I added a controller sensor and a text displayer, but with the text displayer I changed the display to a green triangle. I had the output of the counter that went into the transporter also power on the text displayer and controller sensor. I needed triangle on the controller to execute

two things: destroy the object in the player's hands and emit the object.

I connected it to a destroyer to get rid of the object and since I wanted the object to emit from the player I went into the player's microchip, I added a node and an emitter. In the emitters properties, I needed to choose an object to emit, but once an object is selected it becomes invisible in the scene until it is thrown. What I did was copy the pillow with its logic and selected the second pillow to emit. By having the same logic in the second pillow it made it so the pillow could seamlessly be picked up and thrown.

I set the emit direction to forward from the baby's chest. I did not want the baby to be able to throw the pillow very far, so I messed around with the emit speed and ended up at 0.6 m/s. I also set the emit mode to once since I only wanted the player to be able to throw one pillow when he/she picked it up. Just as a precaution I set the max emitted at once to one, as well.

The next step was creating the cutscene for when the player first interacts with the pillow. I had already done the logic for the latch, so I applied the same for the pillow. With all the gadgets in place, I just needed to create the cutscene in the timeline. I created keyframes to hide the player puppet and to unhide the dad puppet. I stretched them for the entire timeline.

I added a camera gadget and positioned it in the center of the pillow. I changed the transition to cut. I created a few keyframes for the pillow. The keyframes moved the pillow from the bed into the air a few inches to signify the father picking the pillow up.

I then made a few keyframes where I moved the father to the head of the crib and I scoped into him and moved his torso and limbs to make it look like he was picking up the pillow. I aligned the keyframes with the ones from the pillow to better sync the action. At first to make the shaking motion I used different camera gadgets, which had different fields of view, but I replaced them with a camera shaker gadget. I adjusted the camera shaker's settings to 60% shake strength and 20% shake speed. I think



this method worked much better than the different cameras with different field of views.

Since I did not need the a cutscene in the cloned pillow, I could keep the logic that I had cloned, but I did still need to add a selector to hide the square text displayer when the character was holding the pillow. I attached a wire from the counter to move to next selector and sent output A to the power of the text displayer. With that I was finished with the logic for the pillows (see figure 11 for logic).

## THE BLANKET

I needed to create one more item to complete the three objects for the crib, and that was the blanket. I went into sculpt mode, modified the cube, so it was a thin rectangular prism and I set the looseness to twenty-three percent, which made it look more like a blanket. I then used the sphere shape and changed it to look like a pellet. I set the pellet to subtract and contoured the blanket a little with it.

I placed a microchip on the blanket. I already had done all the logic I would need on the pillow, so I copied it over to the blanket and modified what I needed. I changed the pickup to circle and the text displayer to represent a red circle.

I had to adjust the orientation of the transporter, but I still used the same tag. In the player's microchip, I copied the emitter, cloned the blanket and selected the clone as the object to emit. I used the rest of the same settings for the emitter.

Being that I did not need a cutscene in the clone's logic, I simplified it, like in the clone of the pillow. I still needed to make the cutscene for the object though. In the cutscene I wanted to make the blanket go over the sleeping baby. First, I deleted all the items that were in the timeline from the pillow, then I made two keyframes, which were the blanket being moved up. I added a camera gadget and adjusted the view to be looking down.

I tried moving the player puppet to look like it was sleeping, but the puppet would not act how I wanted



Figure 11a. Pillow logic



Figure 11b. Emitted pillow logic



Figure 11c. Emitter logic

it to act. I would place it in certain position, but when I played the cutscene she was in a different positions, so I decided to clone the player puppet and delete all the logic in it. I set the puppet to non-, -collidable, or -visible, but used a keyframe to turn on her visibility and position her in a proper way, so it looked like she was sleeping. I created another keyframe where I hid the player puppet for the cutscene.

## EXITING THE CRIB

I had completed all the objects I wanted to do for the crib, but I still needed to make it so the baby could not jump off the crib until she threw the pillow down. I started by making a wall in sculpt mode. I placed the wall in front of the crib and made it non-visible.

It worked to prevent the character from jumping down, but a problem I ran into was it would collide with the door and other objects; I only wanted it to collide with the player. I went into the walls properties and made it only collidable with friend, I then went into the puppet's properties and labeled her as friend. The gate and other objects could then pass through the wall without any hassle.

To make the wall disappear when the pillow was thrown down was simple. I snapped a microchip into the wall and placed a trigger zone, which needed to detect the pillow, so I made a tag called, "Pillow", and placed one in both of the pillows. I set the trigger zone as a prism around the ground in front of the crib. I made sure it was not too tall to where it would already detect the pillow while it was still on the bed. I connected the zone to a destroyer, so when the pillow is thrown down it destroys the wall and the character can jump down.

I decided it would be nice to give the player a hint about throwing the pillow down. I set up a trigger zone when the character was close to the edge that would display text. I copied one of the other text displays and changed the text to state, "If only there was something fluffy to jump onto". I positioned the

text, so it was a decent distance from the crib.

Seeing that it was possible to throw the blanket down, I wanted to display a different text if the player threw the blanket down instead of the pillow. I already had a trigger zone set up to detect the pillow on the ground, so I copied it and changed the target "Blanket". I created a "Blanket" tag and put one in both the blanket and the blanket clone. See figure 12 for text displays.



Figure 12a. Pillow prompt



Figure 12b. Pillow prompt after blanket thrown

I connected the player zone to output into the blanket zone and when that happened, the blanket zone would send a signal to the new text displayer; it would also send a signal to a counter, with a target value of one, which would output into the selector's move to next output. The A output of the selector was connected to the power of the first text displayer. I used a counter because without it, every other time the character would trigger the zone it would turn the first text displayer back on (see figure 13 for logic).



Figure 13. Barrier logic



Figure 14. Subtracted area for door

## SECOND STAGE

I had created all the assets I needed for the first stage and I was ready to start the second stage of my level. The three objects I needed for this stage was a trashcan, a stool, and an activity table. Before I sculpted these objects and created their logic, I decided it was time to design the actual room. I already had a simple sketch of what I wanted in the room and how I wanted it to look.

I began by creating the walls. I turned on grid snap and stay upright. I went into sculpt mode and made a white rectangular prism. I stamped it and then placed the same prism perpendicular to it, stamped it, and so on until I had four walls to make a room around the crib. I then scoped out and made a much smaller brown prism. I stamped it on the bottom of each wall. Those would be the baseboards of the room.

I wanted a door for the room, so I scoped back into the wall and used the cube shape to subtract an area around the wall which would be the door frame. I scoped into the baseboards and did the same (see figure 14). For the door I scoped out and created a rectangular prism a little smaller than the size of the opening in the wall and stamped the shape in there. I bumped the doors original color saturation up to 100% to make the door a brownish color.

The door needed a handle, so I scoped out of it and selected the filled donut shape to make the rose. I messed with the dimensions, so it was in proportion to the door and I placed it on the door where a handle typically would be located. I went to the finishes option and set it to metallic since I wanted the door to be metal. I then selected the cylinder shape and created the shank, then lastly I used the curve shape to design the door handle (see figure 15).



Figure 15a. The rose and shank



Figure 15b. The handle



The room needed windows, so I scoped back into the wall and used the cube shape to subtract three places in the wall for the three windows. I scoped out and created a rectangular box that fitted into the holes I had made, and colored it brown. I made a brown t-shape in the middle.

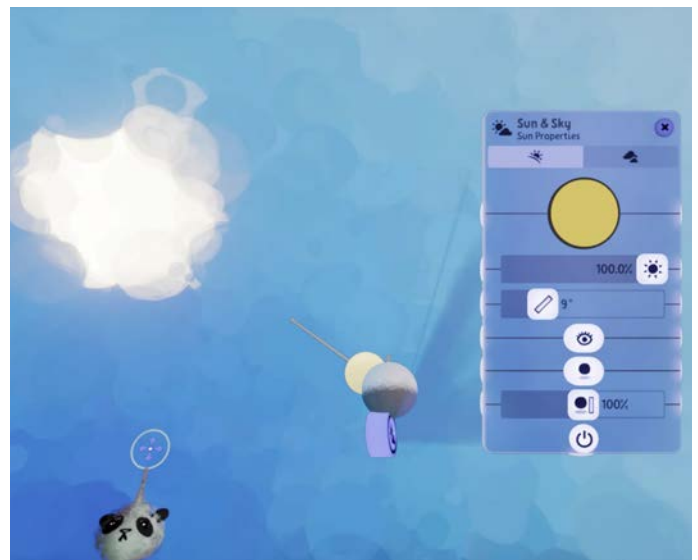
I then went into paint mode, selected a blue color and painted in the for empty squares in my frame. These would be the window panes. I went to the paintings properties and turned the opacity down to 24%, which made it look more like glass (see figure 16). I grouped all these objects together and copied them and placed the two copies in the other two openings in the wall.



**Figure 16. Window pane properties**

To add the ceiling to the room I used the cube shape and modified it to a rectangular prism large enough to cover all the walls. I switched the flecks to straight and stamped my object. With the ceiling done, it was time to add a sun & sky gadget to make the light right for the scene. I put the gadget in the scene above the ceiling. I tinkered with the settings. I turned brightness to 100%, changed the sun size to 9 degrees, increased the sky brightness to 51%, and I changed the angle of the sun, so it would be shining through the windows (see figure 17).

My room needed a light. I went back into sculpt mode and selected the cylinder shape and chose the shiny metal as the finish. I stamped it in the center



**Figure 17. Sun gadget properties**

of the room on the ceiling. I then went to the sphere shape and selected the shiny wax finish. I changed the sphere, so it was very flat and placed it in the center of the cylinder I had made. This would be the light bulb.

Since I had a light, a light switch was in order. I scoped back into the wall and place a rectangular prism by the door hand, which would be the plate cover. I used the cube to subtract a rectangular prism, I then added a rectangular prism in there perpendicular to the plate to represent a switch.

After the light switch and light, I added a carpet. To do this I simply went to sculpt mode and used the cube shape to make a rectangular prism under the crib. I chose the color to be pink-ish and bumped the looseness up to thirty percent to make it look more like a rug.

## ***THE CHANGING STATION***

The next thing I wanted to create was the changing station, which the baby would need to climb up. I selected the cube shape, reshaped it to a rectangular prism and changed the color to blue. I stamped it and then made a smaller rectangular prism which I used to subtract four symmetrical slots in the front for the drawers. I scoped out to make another rectangular prism.

I made a slightly smaller prism and subtracted it from the other prism to make one of the drawers. I then made a thin, slightly larger prism, which was the face of the drawer. After that I used the filled donut shape, with a shiny metallic finish, to be the base of the drawer knob. I used the curve shape, same finish, to make the nob and place it on the base.

I placed the shelf in the changing station and I cloned it three times. I put the clones in the other empty slots, but I had the top right one perturbing, because that was the one the baby would climb onto. Since having all the objects in the changing station set to movable would apply gravity to them and they would not rest how I wanted them to, I set everything to non-movable. I made a pink rectangular prism, with 100% loosens and place it in the open drawer to mimic clothes. I sculpted a rectangular beige shelf above the changing station. This would be where I would place the stuffed animal later.

## THE ACTIVITY TABLE

Now that I had given the room some more dimension, it was time to sculpt the objects with which the character would interact. I began with the activities table. I chose the filled donut shape, changed the color to white, adjusted its dimensions and stamped it.

Next, I wanted to put legs below it. I changed the shape to the curve and the color to blue. Under guides, I turned the kaleidoscope on and selected four (see figure 18). I positioned them under the tabletop and stamped them, I then turned the shape to subtract, adjusted the size and subtracted on the inside of the legs to make the legs look a little more realistic.

I used the wedge tool to design the triangular buttons on the table that would light up. Once I had the shape, I cloned and placed them perpendicular to each other, so they looked like a square cut into four diagonally. I changed the colors of each of them.

The colors were yellow, blue, red, and green. I set the table to non-movable.



Figure 18. Activity table legs made with kaleidoscope on

The design was done, I needed to add the logic. I snapped a microchip onto the table and began. The order was a little different with this one because I did the logic for the timeline of the object's perspective first. I cloned the first section of logic from the pillow. Everything was the same, except I needed to adjust the trigger zone and add a new timeline.

I placed a camera gadget in the timeline and adjusted it, so it was facing up towards the ceiling from the tabletop. I changed the transition type to cut. I made the playable character non-visible. The next task I did was position the father in a sitting position with the baby, but for some reason whenever the cutscene would actually play his legs would be shooting upward and I could not figure out why.

I had made him non-collidable and made him not movable in the keyframes. I could not figure it out, so I made a quick fix. I raised the table, so he was standing and his legs would not do that.

I added keyframes of the different lights on the table illuminating and then I added keyframes of the dad following pressing the sequence that flashed. I had to mess with the keyframes a little to sync the dad's gestures with the flashing buttons. When playing back the timeline, it was a little too fast, so I slowed the speed to seventy percent.



Originally, I was not sure what I wanted the activity table to do when the player interacted with it again. I first used a randomizer and selector to have it light up the different buttons at random, but I decided it would be better to have it play a sequence. I had the second AND gate output to a counter as well as a timer. The counter was set to one and outputted to a timeline and the timer was set to five seconds and would reset the counter. This way the timeline would be replayed after every five seconds. I used the same keyframe sequence of the lights illuminating as the other timeline (see figure 19 for activity table logic).



Figure 19. Activity table logic

## THE TRASHCAN

For the trashcan, I selected the cylinder shape and changed the color to silver. I made a slightly smaller shape and subtracted it in the large cylinder to hollow the can. I then scoped out and used the cylinder shape again to make the lid of the trashcan.

I placed the lid onto it and then I went into gadgets and selected the bolt connector. I connected it in the back from the bottom cylinder to the top cylinder. I adjusted the rotation direction the right way, and put the limit on at 90 degrees. I added a black rectangular prism to the back of the can where the bolt was located. Lastly, I sculpted a couple of white rectangular prisms together to make a little trash pile.

I snapped a microchip to the lid and used the same logic as the activity table to make the cutscene. I created a keyframe that made the player puppet invisible and the dad visible. I then added the camera. I made it so the camera view was facing the inside of the can and would open with the lid.

I proceeded to make the keyframes of the lid opening and then shutting. After that I made the keyframes of the dad walking to the trashcan and moving like he was throwing away something. Next, I added the keyframes of the trash in the dad's hand and being thrown away (see figure 20).

A problem I ran into with this timeline was that the father's feet would go through the trashcan in the first keyframe. I am not quite sure why since he was placed far enough away where that should not have happened. The solution I used for this was I made a keyframe to hide his feet for the sequence.

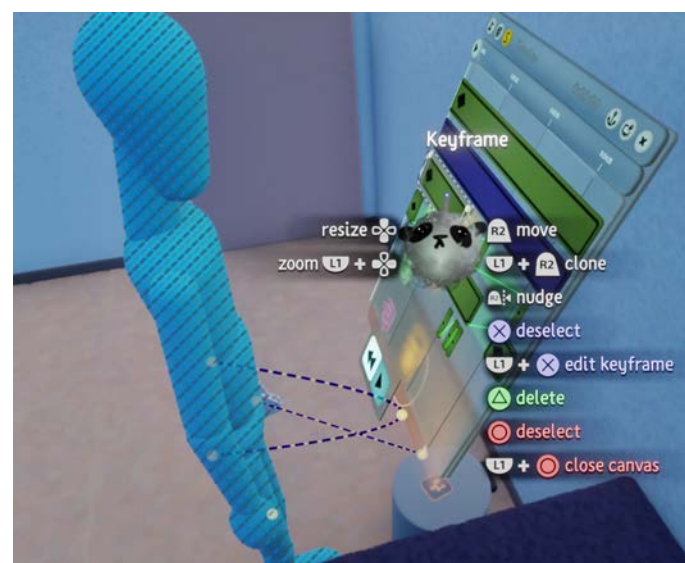


Figure 20. Puppet's movements in timeline

I wanted the player to be able to open and close the trashcan after the cutscene played. As before, I used the same logic to connect to a second AND gate, but I needed to output it to two counters. I set the target value of the first timer to one and outputted it to a keyframe of the lid open. I set the second counter target value to two. I then had its output restart the first timer and restart itself. With this logic the lid would stay open until the player pressed circle again (see figure 21 for trashcan logic).



**Figure 21. Trashcan logic**

## THE STOOL

The last object I needed to complete the second stage was the stool. I created it by first stamping a beige filled donut shape. I used the curve shape to make the legs, I modified the shape, turned the kaleidoscope on and set it to four. With the legs in place, I wanted to make the cross-sections. I created a rectangular prism that went across from one leg to the opposite leg and stamped it, then I did the same with the other two legs.

Again, I used the same logic here as before for the cutscene. I changed the output of the controller sensor and the text from the text displayer to square. In the timeline I created a keyframe which made the stool and playable character invisible and the dad visible. I then snapped a camera gadget in there and adjust the camera so it was facing up. The last keyframes I made were of the dad walking to the stool, picking it up, moving it, putting it down, and then sitting on it. In these keyframes I had the stool follow his actions.

I used similar logic, like for the blanket and pillow for the character's interaction with the object after the cutscene, but it was a little different since I did not want the character to throw the object; I wanted her to drop the object. I still had the second AND gate that went to a counter, which outputted to the text displayer and teleporter, except this time

when the player would press triangle, it would reset that counter, hence turning off the teleporter and dropping the object. Pressing triangle would also move to previous output on the selector so that the text displayer would show square again.

Since I wanted the player to use the chair to climb up onto the open drawer, I had to add extra logic. I realized that in order to climb the chair, I needed to detect two things: the player, and the stool. I snapped two trigger zones to the changing station. I had one detect the "Player" tag and I created a tag in the stool microchip labeled, "ChairTop". I placed the location of the tag right on the top surface of the stool. I set both zone shapes to a cube and adjusted them around the open drawer.

I had both of the trigger zones go into nodes in the stool's microchip and then into an AND gate. I also connected the output from the counter that output to the teleporter to a NOT gate which went into the AND gate. The NOT gate made it so the AND gate it went into was not active, unless the player was not holding the stool. I had the AND gate output to a text displayer, which displayed a blue cross.

A problem I ran into here was that when the stool was in the proper location next to the open shelf, the text displayers would display both square and cross. I needed to find a way to hide the square. I decided that an XOR gate should work to fix my problem.

This gate works by outputting a signal only if one of the inputs is true. Before I had the trigger sensor for "PickupDetection" output directly into the square text displayer, I put the XOR gate in between the two and had the AND gate that output to the cross text displayer also output to the XOR gate. This would stop the signal to the square text displayer when it was showing the cross (see figure 22 for logic).

I had solved the issues with the text displayer and the next thing I needed to do was add the logic for the puppet to climb up the chair. I placed a microchip inside of the puppet's logic and called it, "Climb Up Stool". I placed three nodes in the microchip, two which received the two trigger zones snapped on



**Figure 22. Stool logic**

the changing table and one from the cross button on the controller sensor. All three nodes went into an AND gate which outputted to a teleporter. I selected the, "ChairTop" for where the puppet should be teleported, turned match target position on, and adjusted the location of the teleporter on the puppet so she would teleport to the correct location (see figure 23 for logic).

The teleporting was working fine, but the stool would easily topple over when the puppet teleported onto it. I needed to find a solution. I attempted to sculpt in a rectangular base on the bottom of the stool, make it non-collidable, and hide it, but that did not work. I tried making the chair really heavy, which worked to teleport on, but the puppet could not carry it then.

It would cause the character to slide across the floor when holding it.

I resolved to making a keyframe, which would make the stool as heavy as possible when activated. I only required the stool to be heavy when the character put it down, so when the character pressed triangle to put the stool down I also had triangle output to a counter that went to one which powered on the keyframe. I also connected the output of the square button to restart the counter so that the character could still pick up the stool again.

The last part I needed to do was add the logic to get the puppet onto the drawer. I snapped a microchip onto the pink box I had placed in the drawer earlier. In the chip I added a trigger zone with a cube shape and had it target the "Player" tag. I had to adjust the dimensions of the zone so that it would only pick up the player if the player was on the chair.

I had the zone output to a text displayer, which displayed a circle over the open drawer. I created another microchip in the puppet labeled, "Climb Up Shelf". I outputted the trigger zone to a node in here and had outputted circle from the controller sensor in the puppet to another node in this chip.

Both nodes, like with climbing up the stool, outputted to an AND gate which outputted into a teleport. I placed a tag in the drawer chip called, "Drawer" and moved its location slightly above



**Figure 23. Climb stool logic**





**Figure 24. Climb shelf logic**

the pillow. I cloned the teleporter from before and changed it to target the “Drawer” tag ( see figure 24 for logic).

In testing getting on to the changing station, I notice the character was not able to walk on the table because the surface was too high. To fix this I used the wedge shape to create and incline. With the slight elevation of the wedge, the character could walk onto the table. I made the wedge invisible and not movable.

## **FINAL STAGE**

I came to the last stage of this demo. There were four objects I needed to sculpt and program: the baby monitor, desk lamp, ball, and stuffed animal. The first object I started with was the baby monitor.

## **THE BABY MONITOR**

In sculpt mode I selected the rounded cube, I made it white, and adjusted the dimensions. I sculpted the filled donut shape above that to give it a round top. I then selected the cylinder and stamped that on the right side as the antenna. I sculpted a black sphere on the top.

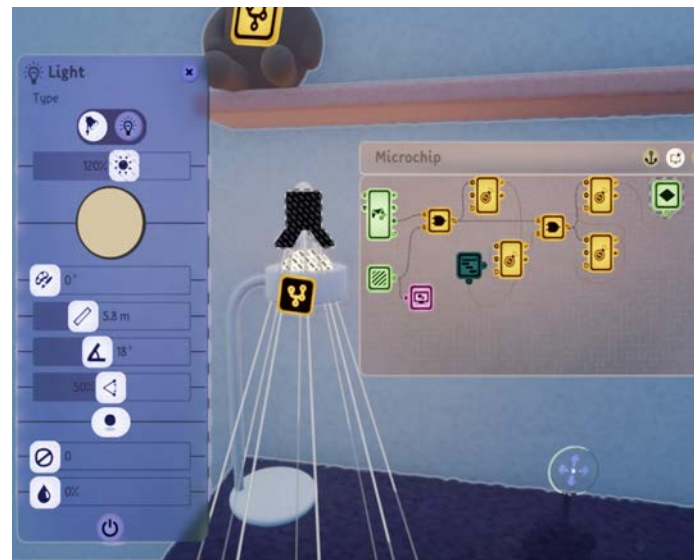
I then made and stamped two horizontal oval spheres next to each other, close to the base of the monitor. The left one was red and the right green. I used the cylinder to subtract an area on the bottom right side of the monitor and then stamped a small black cylinder in there to represent a volume dial. The last thing I did was use the donut shape to subtract a ring around middle and top of the front to represent a speaker.

The logic for the baby monitor was exactly the same as the activity table so I cloned the microchip and snapped it onto the monitor. I adjusted the dimensions of the trigger zone and text display. I went into the timeline for the cutscene and deleted everything except the keyframes that made the character invisible and the dad visible, and I adjusted the view of the camera so it was looking forward out of the speaker of the monitor. I then made some keyframes of the father walking towards the crib and consoling the baby.

I wanted a static noise to play for the character interaction with the baby monitor after the cutscene. I deleted the second timeline that I copied from the activity table and had the counter output to a static noise I found on the Dreams search tool. I made a timer that was activated when the noise was and would reset the counter after three seconds (see figure 25 for logic).



**Figure 25. Baby monitor logic**



**Figure 26. Light logic**

## THE LIGHT

The next object I moved onto was the desk lamp. I started off by creating the base, which was a thin horizontal white filled donut shape that I stamped. I then stamped a thin vertical white cylinder on a top edge of the base. I used the curve shape to create a 45 degree curved cylinder and stamped it to the top of the vertical cylinder. I stamped a larger horizontal cylinder to the end of the curved cylinder, this would be where the bulb would go later.

I needed to hollow out the cylinder that would hold the bulb and I also wanted the top to have a bit of a lip. I achieved this by using the cylinder shape and subtracting inside of the cylinder. I did not subtract through the whole cylinder so the top was still closed. I used the filled donut shape and subtracted a smaller diameter of the top section of the cylinder. This gave the top the little lip around the opening, which I wanted.

I created the bulb as a separate object. I sculpted a white sphere with a shiny finish. I then used the cylinder shape to subtract almost half of the bottom half of the sphere. To create the lighting effect, I snapped a light gadget to the bottom of the bulb. I adjusted the light so it shone through the bulb downwards. I changed the beam range to 5.8 meters and the beam angle to 18 degrees (see figure 26).

I found that I liked the brightness the best at 120%, but I turned it down to zero because I would have it change to 120% in a keyframe. I snapped a microchip onto the light and cloned all the logic from the trashcan since the function for both was the same: open/close, on/off. I altered the location of the circle display and the zone for the trigger zone.

I deleted all the keyframes in the cutscene timeline, apart from the keyframe that made the player invisible and the dad visible, and the camera gadget. I changed the location of the camera to the bottom of the lamp bulb and had it face down. I created some keyframes where the father's hand went to the base of the desk lamp and switched on the light.

After that I created a keyframe for when the dad turned on the light, with the brightness of the light switched to 120%. In the last keyframe, I modified the settings for the sun and sky to two percent so it was dark. I also turned on the glow of the ceiling light to thirty-three percent.

## THE BALL

The ball for the player to throw was simple to sculpt. I just selected the sphere shape, adjusted the size and made it white. I placed it at the front left edge of the changing station. The logic for the ball was exactly the same as the pillow so I created a microchip on the ball and cloned the pillow logic into it. I adjusted

the square display location and the trigger zone area and went onto the cutscene timeline.

As before, I deleted all the keyframes except for the frame which made the dad visible and the player character invisible and I changed the camera so it is facing down from the bottom of the ball. I added many keyframe where the dad was tossing the ball up in the air multiple times. The playback speed was too fast when I tested it so I changed it to 100%.

The logic to pick up and throw was the same, but I wanted the ball to be in the character's hand when the player picked it up. I scoped into the player puppet and snapped a tag on her hand labeled, "Hand". I then went to the teleporter and selected it to teleport to "Hand" (see figure 27). I adjusted the teleporter so that it was in the center of the ball .



**Figure 27. Hand tag**

Like with the other objects that the player could toss, I needed to attach the ball to an emitter in the player's microchip. I did that and cloned the ball. I went into the emitter's properties and selected the cloned ball as the object to emit. I messed around with the speed and angle of the emitter and ended on 3.9 m/s and a directional arrow around a 60 degree upwards slant. I then went into the cloned ball and deleted all the logic for the cutscene.

## STUFFED ANIMAL

I came to the object I needed to create to fulfill the objective, the stuffed animal. I went into sculpt mode and selected the filled donut shape. I changed the flecks to streaky and the color to a dark brown. This was going to be the body of the stuffed animal.

Since I wanted to attach all the limbs and head with connectors, I sculpted them all in separate groups. I used the curve shape to create the left arm and placed it into position. I cloned the arm and then removed it from the group and reflected it to create the right arm. I used the same process for the legs.

The head was a bit of a different process. First, I created the base of it, adjusting the sphere shape into a horizontal oval shape. I then used the filled donut shape to create both ears. I went back to the sphere shape and created a beige ovular sphere for the snout. I reduced the size and used the shape to make a small black nose. Lastly, I used the sphere to make two small black eyes.

I attached the head to the body by using two bolt connectors. I limited the angle to 180 degrees and adjusted the rotation from the front to the back. I attached the arms to the body with the same connectors and limit, and made it so they could rotate back and forth. I used the same procedure for the legs, but set them in a sitting position. I placed the stuffed animal on the shelf above the desk lamp.

The stuffed animal required some logic so I snapped a microchip onto it. Since I wanted the stuffed animal to fall down, I put an impact sensor in it. When an object would hit it, it would output a signal.

If bumped, the impact sensor would output to a counter set to one which would output to a timeline. In the timeline I made keyframes of the stuffed animal falling from the shelf onto the changing station. Since I wanted the stuffed animal to stay at the bottom after the timeline played through, I set the last keyframe to keep changes.



I wanted the character to pickup the stuffed animal after it fell so I used the same logic as before to pickup. I edited the trigger zone and text displayer location, and I set the teleport to the “Chest” tag. I had to adjust the axes of the teleporter to have it teleport with the proper orientation on the puppet. Being that this was the end of the level, I did not add the code to drop the stuffed animal (see figure 28 for logic).



Figure 28. Stuffed animal logic

## GAME SCENES

The character actions in my scene were complete, but I still needed to add the opening and closing video, as well as the camera panning to the stuffed animal when the character jumps off the bed. I began with the opening scene. Since it was not related to any object specifically, I just placed a timeline into the middle of the scene. Since I did not want the timeline to be active whenever I went into play mode to test the game, I attached a switch to it so I could turn it on and off. The first thing I placed in the timeline was a keyframe to unhide the dad and hide the player puppet.

I snapped a camera gadget onto the timeline and placed it where the baby’s head would be and had it facing upwards. I wanted the camera to pan down to the stuffed animal and then zoom in on it. In order to do this I needed a few camera gadgets. I had three camera positions on one line. These represented the beginning view, the view when the baby looks

down to the stuffed animal and the view of when the camera zooms in all the way.

There were two gaps between these three views, I copied the second and third, and underneath placed them filling this space. The top cameras were set to cut, but I set the bottom to smooth transition and the transition time for the first to two seconds and the second to three seconds. This was done to give smooth movement between the different camera views (see figure 29).

In the next step I added a great deal of keyframes of the father tucking the baby into bed. With that the opening scene was done, but I did not like how it would go directly to the camera looking up and then cut quickly to the game, I wanted the screen to be black and ease in and out of the cutscene. I did this by adding two wipers to the timeline. I set the transition color to black. I made the fade-out time 5 seconds for the wiper at the start with a fade-in time of 3 seconds, and set a fade-out time of 1 second for the end wiper.

The logic to look at the stuffed animal when the character jumped off the bed was pretty simple. I snapped a microchip onto the floor around the area where the character would jump. In the chip I placed a trigger zone and had it detect, “Player”. I set up a square zone around the area next to the crib.

I connected the zone to a timeline. In the timeline I made a keyframe to make the character non-visible and I added two camera gadgets. One camera faced the stuffed animal and the second one was zoomed in on the animal. For the second gadget, I set the transition to smooth and the time to three seconds to blend between the camera views.

The transition between the cameras was fine, but I was having issues with the transition from the gameplay to the first camera. It would go into the floor and then float up to the position I wanted. I fixed this by changing the transition to linear and also having a time of three seconds.

The ending cutscene happens after the character picks up the stuffed animal so I went back into its



**Figure 29. Intro cutscene timeline**

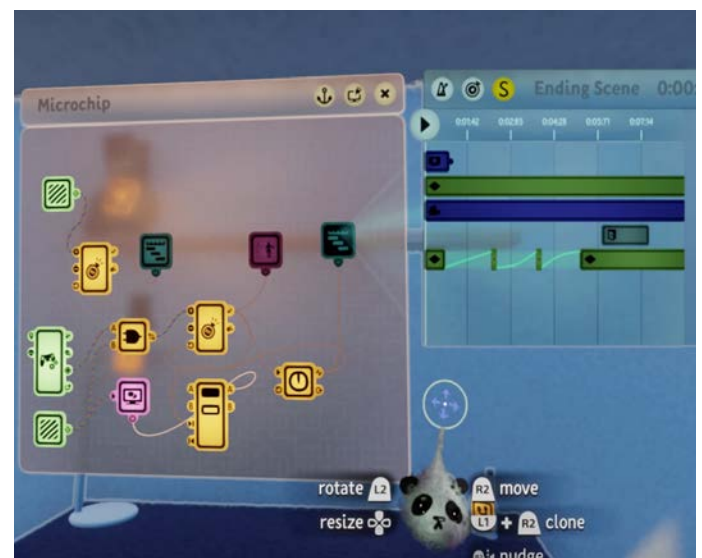
microchip to add the ending timeline. I did not want the timeline to play right after the player picks up the stuffed animal so I placed a timer before it set to 1.5 seconds. I used the counter that activated the transporter to also activate the timer.

In the timeline, I created a keyframe to hide the player puppet, unhide the non-playable girl puppet, hide the blankets, cloned pillow, and placed the other pillow in the proper position in the bed. I created a keyframe that put the stuffed animal in the bed next to the non-playable baby. I created a camera gadget and placed it behind the stuffed animal in the crib so it looked like the baby's perspective. I had clipping issues with the baby's head so I hid it.

I wanted to show that the baby was happy with the stuffed animal so I created a few keyframes where the baby was rubbing it. The action of the cutscene was complete so I added a wiper to the beginning of the timeline. I set the transition color to black, used a circular wipe and set the fade-out to 1 second. The last gadget I needed to add was a doorway to end the level. I snapped it at the end of the timeline, made the transition color black, circular wipe, and a three seconds fade-in.

A problem I encountered while doing this was initially the stuffed animal would not be in the crib in the cutscene. I realized that was because of the teleporter. It was still active and that would override

the keyframe. I was able to fix this by having the timer output a signal to the counter to reset it. This would turn off the teleport and fixed the problem. I also realized the cutscene would be better if it was a little darker so I made a keyframe where I changed the sky brightness to twenty-three percent and the sun brightness to zero (see figure 30 for logic).



**Figure 30. Stuffed animal logic with ending timeline**

## MODELLING PUPPETS

I had completed a rough demo of my scene, I decided it was time to add a little more detail to my puppets. I started with the player puppet. I scoped into her body and I selected pink spraypaint. I colored her bottom, abdomen, and chest section to



form a kind of onesie. I also selected a beige skin color and spray painted the rest of her body.

I sculpted into her head and used the sphere shape to stamp an ovular shape for the head because her face was too long for a baby. After, I selected the sphere shape and turned mirror on. I subtracted two holes in her head for her eye sockets. I then halved the spheres and stamped them on the tops of the sockets to make a brow. I used the curve shape to make brown eyebrows, I made the looseness kind of high to make the eyebrows more hair-like.

I used the cone shape without mirror to stamp the middle of the nose, and two mirrored half circles for the nostrils. For the lips I mirrored the curve shape and changed the color to a more orange-pink. I scoped out of the head and then stamped two white mirrored spheres in the eye sockets. On both eyes I spray painted black circles for the pupils.

I scoped out of the eyes and sectioned the sphere shape in half and hollowed it. I changed the color to a beige brown, changed the flecks to streaky, and stamped two of the shape on the top and back of the puppets head to make hair. I made sure the looseness was higher and stamped a piggy tail on the top back of the head with the curve shape.

I did a similar procedure with the dad puppet. I scoped into the body and used spraypaint to color his

body with the same skin tone as the baby. I sprayed the torso and upper arms white for a t-shirt, his pelvis and legs blue for jeans, and his feet black for shoes.

The puppet's head shape did not need to be altered, but I did need to add some features. I used the mirror and made sockets for the eyes. I placed eyeballs with pupils in the sockets and used half-hollowed spheres to stamp eyelids over the tops and bottoms of the eyes. This time for the nose I used two mirrored cones to make it.

I used the same two spheres method with loose streaky flecks to make the hair for the top, which I made black. I wanted the hair to have a little more range so I used the cube to make a rectangular prism and stamped that in the front and on the sides of his hair. I used the curve shape mirrored to make some sideburns for him. See figure 31 for a before and after model comparison.

Since I was using a cloned puppet of the baby for the cutscenes, when I added more detail to the player puppet, I had to delete the clone and make a clone of the current player puppet. This meant that I had to redo all the keyframes in the cutscenes that had the cloned puppet in them. I already knew the cloned puppet's actions in each cutscene so it did not take too long to update them.



Figure 31a. Old baby puppet



Figure 31b. New baby puppet

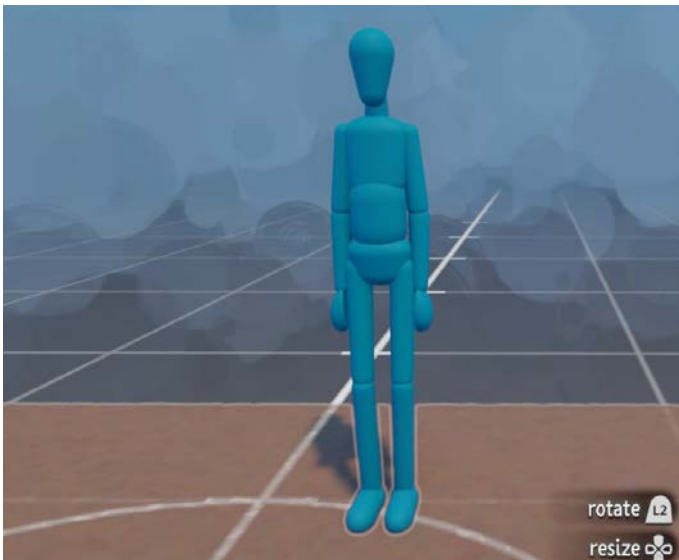


Figure 31c. Old dad puppet



Figure 31d. New dad puppet

## GESTURES

The functional logic was completed for the interactive objects, but I wanted to adjust the puppet's arms for objects she could pickup to make it look like she was holding them in her hands. In order to do this I needed to add a keyframe in each object with the arms positioned to the desired place. The keyframe would be activated by the same counter that output to the transporter. The slow power up would be turned up to three seconds for each keyframe.

I created three keyframes: one for the stool, one for the ball, and one for the stuffed animal. For the stool and stuffed animal, the puppet would be holding the object with two hands and for the ball, she would hold the ball with her right hand. I did not add any keyframes for the pillow or blanket because the objects were so large they covered the bottom of the screen and so the player would not be able to see them anyways.

I already had the text displayers to let people know how and what objects they could interact with, but I wanted to add more visual cues. I did this by adding hand gestures when the player was in the vicinity of an object. The base logic seemed simple enough, I would use the trigger zone to input into a counter with the target value set at one. The counter would output to a keyframe of the puppet reaching out for the object.

The logic worked to turn on the keyframe, but I needed it to turn off when the player walked away and turn on when the player was back in the zone. I achieved this by attaching the trigger zone to a NOT gate and having the NOT gate restart the counter. I put the logic in each keyframe and I made different gestures for the objects. I adjusted the slower power up and slow power down in the keyframes to 1.5 seconds to make the movement of the arms more smooth.

I wanted to have different arm animations for the stool. One animation for when the player needed to pick it up and another animation for when it was by the changing station and the player could climb up it. I went into the player microchip and into the "Climb Up Stool" microchip. In here I used the two trigger zone nodes and output them into an AND gate and output that into a keyframe. The keyframe had the same settings as the one to pick up the stool, but I rotated the hands position so it looked more like they were ready to climb.

I messed around with a few methods to prioritize the climb gesture and I landed on using an exclusive gate gadget. I placed one in between the AND gate and keyframe in the "Climb Up Stool" chip and one between the trigger zone and keyframe in the stool microchip (see figure 32). The gadget's default priority is one so I changed the priority for the climb gesture to two to override the stool's animation. In figuring out the override for the climb gesture, I

realized that I did not use a counter or NOT gate to activate the keyframe, it worked fine by getting the signal direct from the trigger zone. I went back to all the other objects and simplified the logic.



Figure 32. Hand gesture override for stool

## PILLOW SOUNDS

I had almost everything that I needed in this demo, but I was missing sound for most of the scene. I started by adding sound to the pillows logic. The first sounds I added were in the cutscene. I wanted to add two sounds: the dad humming and the pillow being fluffed.

I was not able to find any sounds on the *Dreamiverse*, which fit my needs. I also could not upload any noises from my computer onto Dreams because of copyright issues. My only alternative was to record the noises myself.

I had a pair of headphones, which had a built-in microphone that I could plug into my Playstation controller. I went into sound mode and dropped a sound recorder gadget into the timeline. Once the gadget is dropped, it automatically begins to record. I recorded the sound of me humming. I had to do a few takes to finally get one of which I approved. I edited the length of it to fit the duration of the timeline.

For the pillow fluffing noise, I used one of my bed pillows and fluffed it by the microphone. I was surprised at how well the microphone was able to pick up the sound of the pillow. I trimmed the noise so that it was only the duration of the camera shaker.

I wanted there to be a sound for when the pillow was picked up and thrown. I found a noise in Dreams called, "Bouncy Shoe Drop". I think the noise fit the actions, it was just a little loud so I turned the volume down to forty percent.

I attached the output from the counter that went into the teleporter and other gadgets into the sound gadget. I copied the sound and had the controller sensor output to it (see figure 33). I copied the sound gadget and did the same for the cloned pillow.



Figure 33. Pillow logic with sound

## BLANKET SOUNDS

The next object I added sound to was the blanket. I opened its microchip and went into the cutscene timeline. I wanted two sounds for this cutscene: a noise of the baby sleeping and the movement of a blanket. I tried to find something in the *Dreamiverse*, but I could not, so I first did a long recording of me snoring. I trimmed it down to the best section and repeated it once.

Below it I made another recording of me moving a blanket. Unfortunately with my microphone, I could not pick up the blanket noise that well so I had to do a few takes and crank the volume from the default eighty percent up to one-hundred percent. I used the same "Bouncy Shoe Drop" noise and logic for the blanket and added it to the clone blanket.



## LATCH SOUNDS

Next up, the latch, I only needed two sounds for the timeline: the latch moving and the father humming. I was able to find a squeaky latch noise in the *Dreamiverse*. I placed a longer version of the squeaky latch noise for when the latch first starts to rotate and then I added three small consecutive ones at the end when the latch was open and bobbed. I changed the tempo of the ones at the end to two-hundred percent.

For the humming, I reused the humming I made for the pillow cutscene. The cutscenes were the same length so I did not need to trim or expand it. I cloned the latch noise and also snapped it onto the microchip for when the player opens the latch. It was activated by the output of the second AND gate; the same output which activated the latch opening keyframe.

## ACTIVITY TABLE SOUNDS

I had finished adding sounds to all the objects in the baby crib, I then went to the objects on the ground and started with the activity table. In the cutscene timeline, I added a different beep noise for the different colors that lit up. I lined them up with the keyframes of the lights turning on. To give them different sounds I changed their tempos. I changed the first 42.67%, the second and third to 46.25%, and the third to 49.76%.

After that I added a joystick click synced up with the dad pressing the buttons. I found all the noises on the *Dreamiverse*. I cloned the beep noises and aligned them with the keyframes in the second timeline.

## TRASHCAN SOUNDS

The next object was the trashcan. I went into the first timeline and added a spring latch noise for when the trashcan opened and closed. I changed the tempo for the lid opening to 46.58% and the one for closing to 82.90%. I then added a flappy cloth whoosh that started when the trash, which the dad drops in the can, starts falling down. I changed the tempo of that to thirty percent.

I wanted the same spring latch noise for when the player opened and closed the trashcan. The logic to play a noise was different than the open/close logic. I had an output from the second AND gate go into another counter with the target value of one. The output of the counter activated the noise and also reset the counter (see figure 34).



Figure 34. Trashcan with sound logic

## STOOL SOUNDS

I was finished with the noises for the trashcan so I moved onto the stool. I went to the cutscene and I dropped in a sound of something being dragged between the keyframes of where the dad is moving the stool. I changed the sound to fifty percent and the tempo to 43.67%. I then dropped a sound of a wooden creak during the action of the father sitting on the stool. For this sounds I turned the volume all the way down to twenty percent.

Like with the pillow and blanket, I wanted the stool to make a noise when you pick it up and put it down. I found a sound called, “Huge Wood Buckling”, and I used that for the noise. I powered it on by the same counter that powered on the teleporter. The noise was loud so I turned the volume down to forty percent. I copied the sound and powered it on with the same counter that turned on the keyframe to make the stool heavy.

## **BABY MONITOR SOUNDS**

I had already added the noise to the baby monitor for when the character turns it on, but I still needed to add sounds for the timeline. I wanted two sounds: a baby crying and the father consoling her. I found a clip of a baby crying. I put it in at the beginning of the timeline, adjusted the volume to one-hundred percent, the tempo to two-hundred, and trimmed it to around seven seconds.

I could not find a sound of a person consoling someone so I used my microphone and recorded some noises of me saying, “shhh shhh shhh”. I trimmed the clip down to around four seconds and reduced the volume to twenty percent. I moved the sound so it started a couple seconds before the baby’s crying halted so it seemed like he had quieted the baby and then there was a moment of silence before the timeline ended.

## **LIGHT SOUNDS**

I went into the timeline for the lamp. I just wanted a sound of the click from turning on the lamp. I found one in Dreams and placed it within a split second after the dad’s hand in the cutscene moves to turn on the light. I changed the tempo to two-hundred percent. The logic for the sound when the character would turn the light on and off is the same as the trashcan so I copied it and had the counter output to a clone of the clicking noise.

## **BALL SOUNDS**

The ball was the second to last object to which I needed to add sound. I went into the cutscene and found an impact sound. I placed it under the keyframes when the dad throws the ball and when the dad catches it. I changed the volume to one-hundred percent. The logic for the sounds when the character picks up/throws is the same as the pillow and blanket so I copied that with the impact sound. I reduced the sound here back to eighty percent and added it to the cloned ball, as well.

## **STUFFED ANIMAL SOUNDS**

I was on the last object, the stuffed animal, and for this I needed a few sounds. First, I wanted a sound for when the stuffed animal got hit by the ball. I did this by having the impact sensor also output to the sound of a hit. I bumped the volume up to one-hundred percent. Next, I wanted there to be a noise for when the stuffed animal hit the changing station. I went into the first timeline and added an impact sound to that under the keyframe when it hit the station.

I went into the second timeline, which is my ending scene to add some sounds. I was able to find some audio of ambient noise with crickets, which I snapped in and had start after the wiper was done. I changed the volume to forty percent. I then added audio of an object being rubbed, changed the volume to twenty percent, and cloned it. I trimmed the clone and then synced the noises with the baby rubbing the stuffed animal.

## **OPENING SCENE SOUNDS**

I had all the sounds for everything I needed, except for the opening scene. I went into the timeline and added a clip with ambient neighborhood noises. I had the sound start after the wiper was done and had it play through the clip until the ending wiper played. The sound was loud so I lowered the volume to forty percent.

The second audio I put in there was my own voice that I recorded making gibberish noises. I figured to do gibberish to mimic the baby not understanding language yet. I adjusted the volume to thirty percent and moved the audio to begin when the father was using his arms to tuck in the baby. I trimmed it to end a little before the ending wiper played.



demo |

# ***IMPROVEMENTS***

# TESTING / FIXING BUGS

Testing and fixing bugs is an important part of making any game. The testing itself is not so difficult, but having to fix the bugs you find can become quite frustrating. Most of the things I fixed have already been stated in the previous sections, but I wanted to shed a bit more light on some of the issues I encountered.

A bug that caused me the most stress had to do with the door for the baby crib. In the pillow cutscene, if the door to the crib was already open, it would be open in the scene. I went back into the pillow cutscene and made a keyframe where the door and latch were closed, but it did not work. The gate would almost be closed, but it would be shaking (see figure 35).



**Figure 35. Latch bug**

I messed around with different settings, making the object movable, changing the labels and collision labels, but nothing worked. Making it non-movable meant that the gate would not go back to the closed state. I was very confused and did not understand why it did not work, it should have worked when I made that keyframe.

I went back to the original settings, but kept the keyframe with the door and closed. Eventually, I found out that I had a keyframe in a different object, the baby monitor, which was telling the door to stay open during the cutscene. I de-animated the door in

that keyframe and then the door was in the proper location in the pillow cutscene.

I thought I was done with dealing with the door, but I noticed that if I would open the door and then quickly go to the pillow cutscene, the gate would again be open in it. Since this would only happen if I did the pillow cutscene quickly after, I figured it had to be some sort of timing issue. I changed the timeline for the door opening to 3.33 seconds. The door would then swing really fast open. I changed the playback speed to five percent and that seemed to solve the issues, the door would open slow enough, but it would still be closed in the pillow scene because the open door timeline was completed faster than before.

The door bug was the biggest problem I had with fixing cutscenes, yet I still ran into little problems with objects not being where they were supposed to be in certain cutscenes because the player had interacted with them in some way. One example was the pillow would not be in the blanket cutscene if the player had already thrown the pillow. That was because the original pillow in the scene had been destroyed when thrown so I needed to make a keyframe for the original pillow in the timeline so it would always be there.

The stool was the second most frustrating issue with which I had to deal. When I first tested teleporting the character on top of it, there were some problems with stability, but I changed some settings and I thought I managed to make it work. I noticed later when testing the game, climbing up the stool did not really work anymore. I was not sure if I had changed some settings when I was working on other stuff or not, but I had to come up with a solution. Luckily, as I stated in the earlier section, I did end up fixing it with the “heavy” keyframe.

The logic to throw an object took me much longer than it should have. I could not figure it out. The character would pick up the object and the character could drop the object, but she could not throw it.

I looked at other people's projects, which had throw logic, and tried to figure out what I was doing wrong. I tried adding a new object in their projects, copying the logic from their objects into the new ones, and seeing if I could have their character throw it. Their character would not, it would always emit their object.

It was not until I noticed I was making the simplest of mistakes, I was not selecting my object to emit. Once I realized that, I did not have any issues with throwing objects. I could copy the same logic, I just needed to select a clone of the chosen object to be emitted.

When I was playing the game, something I noticed, after cutscenes would play, is that the POV would not cut instantaneously back to the character, the camera would sort of glide back into the character's POV. I realized this must have been because of the camera settings I had for my character. I went into my character's camera settings and I changed the transition type to cut and that fixed the problem.

Another issue I became aware of during a play-through was that the slats in the baby crib were not properly aligned. I scoped into the slats and turned on the grid and stay upright. I moved them so they were properly aligned to the cross-boards, but they were a little short so I used the stretch tool to make them longer.

I noticed when throwing the ball that it did not have much bounce to it and it would stick sometimes to the sides of objects, like the desk lamp or changing station. I reduced the friction to ten percent and the increased the bounce to forty percent. I saw that the ball would still stick to the side of the changing station so I changed the friction of that, the desk lamp, and all other objects it could stick to ten percent and that solved the issue.

There was an issue with the timeline for when the baby jumped down from the crib. The timeline would play every time the trigger zone was activated. This was fixed by placing a counter, set to one, between the trigger zone and the timeline.

A little oversight I noticed later was the color of my emitted objects. I had changed the color of all the objects that I cloned to throw so I could better understand the logic. When I was testing my game, I realized that I should probably change them to the correct color so that they seamlessly blend in with the objects they were mimicking.

## **POLISHING LOGIC**

In the process of writing about all of the logic for this project, I would go over everything, and it helped me clean it up a bit. Again, I mentioned most of the instances earlier, but here are some more examples. The crib was the first section I worked on so when I went back to it, I saw things that could be cleaned up. Initially, I had the text displayer snapped onto the inside of the crib door; I moved it into the latch microchip and adjusted the text displayers position. I also had another chip on the latch with some of the logic so moved that into the aforementioned chip.

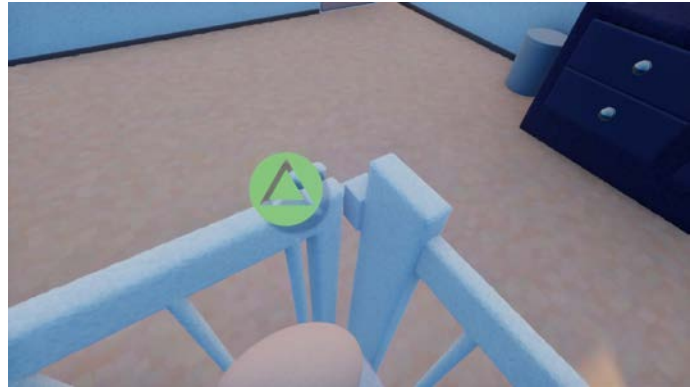
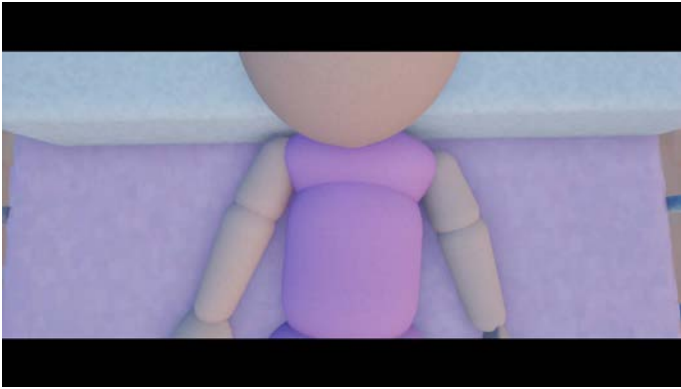
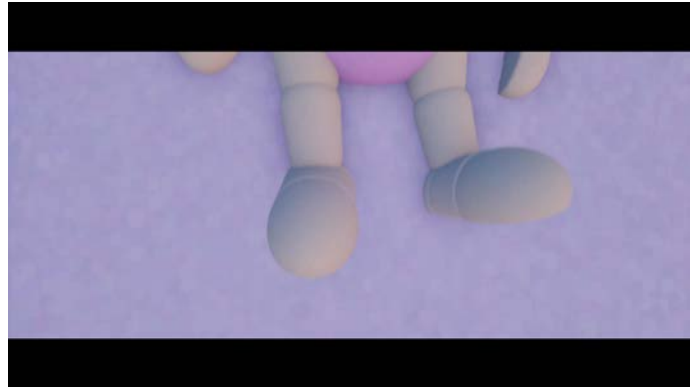
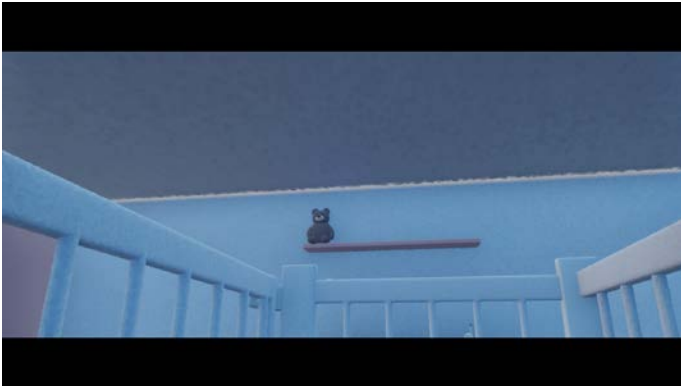
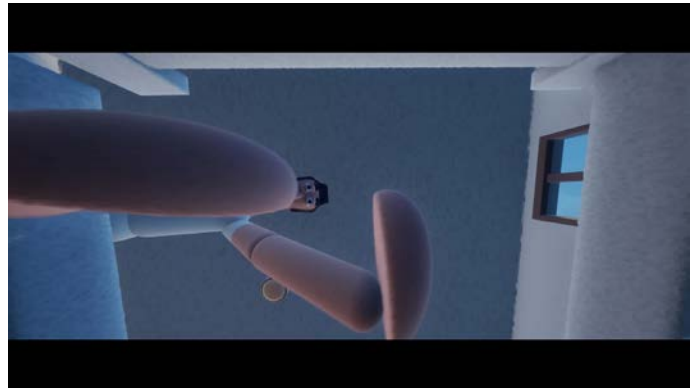
When I first did the logic for the stuffed animal, I had a trigger zone set up around the animal. When the player threw the ball it would detect the "Ball" tag and output to the timeline of the animal falling. Later, I realized that it made much more sense to use the impact sensor since it was designed for that function.

Looking at all the different trigger zones I had, I saw that some were set to detect the "Player" tag and others were set to detect possessed controller sensors. I wanted to consolidate the zones for the players so that everything was unified. I changed all the detect controller sensors to detect "Player".

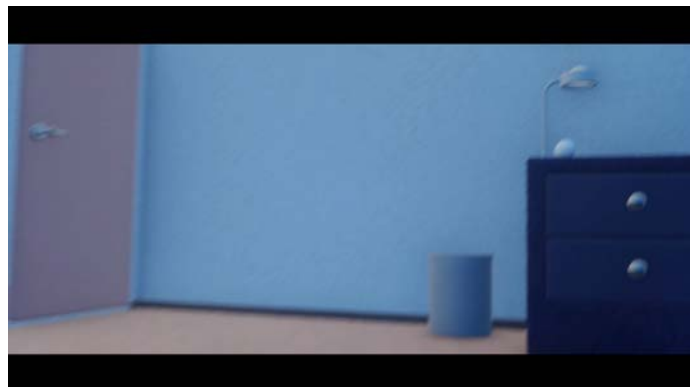
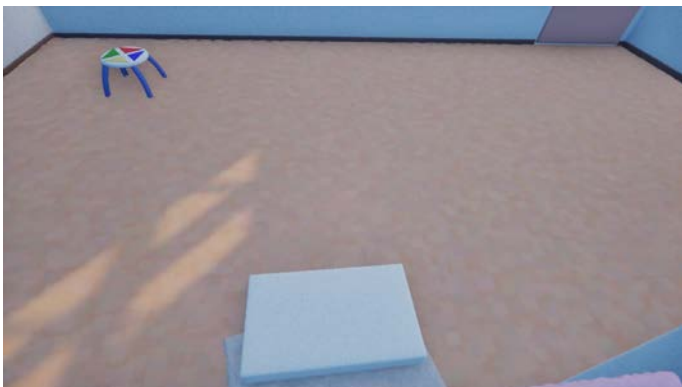
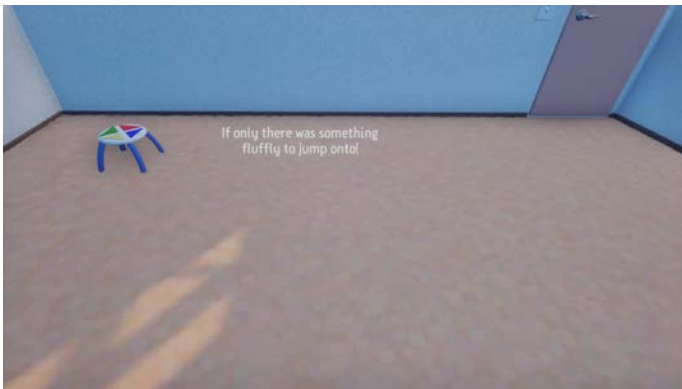
Sometimes counters are needed if you want a constant signal being sent out after the target value is hit, but it is not always the case that you need one. Originally, I had used a counter to activate the keyframes for the hand gestures and I needed a NOT gate to turn off the keyframes. I remembered that a trigger zone would send a constant signal whenever it was triggered so all I needed was to have the zone output directly into the keyframe.

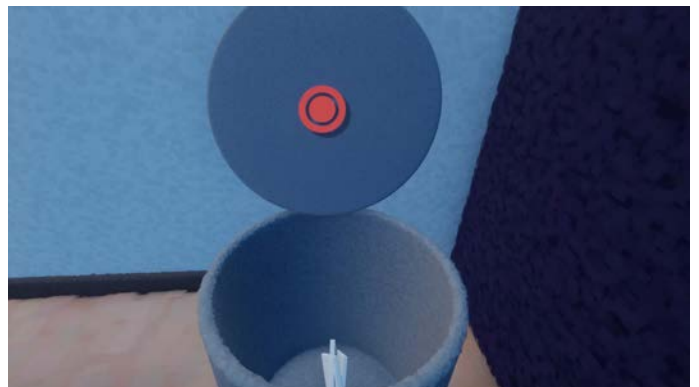
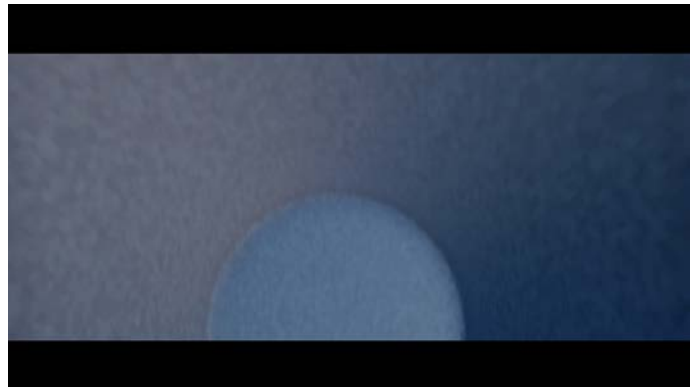
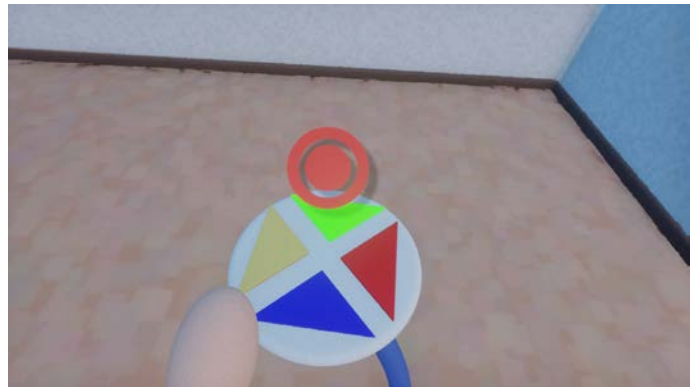
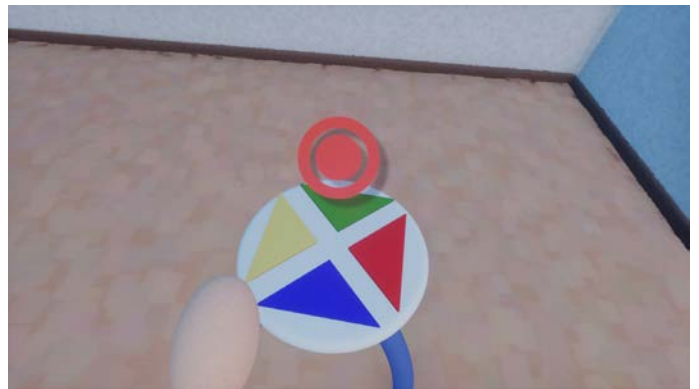
demo |

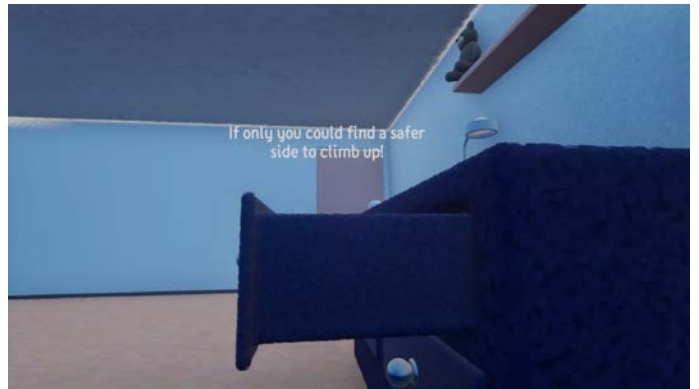
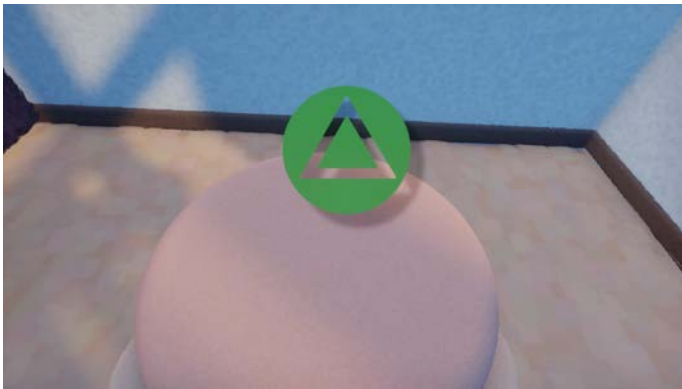
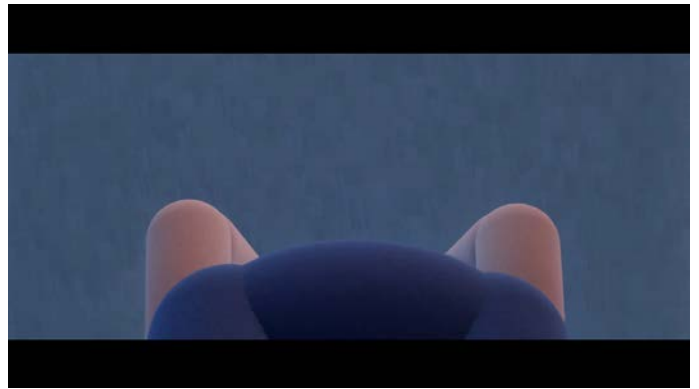
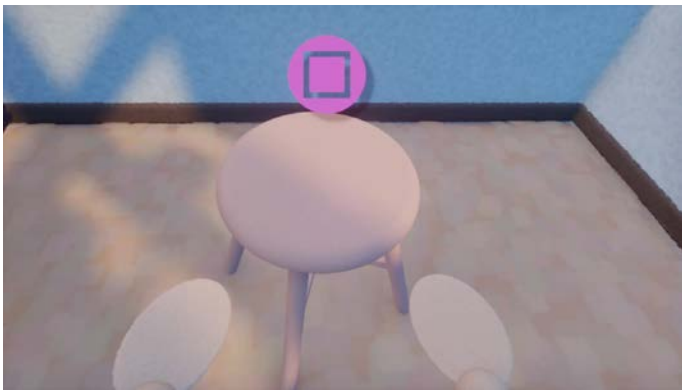
***PLAYTHROUGH***



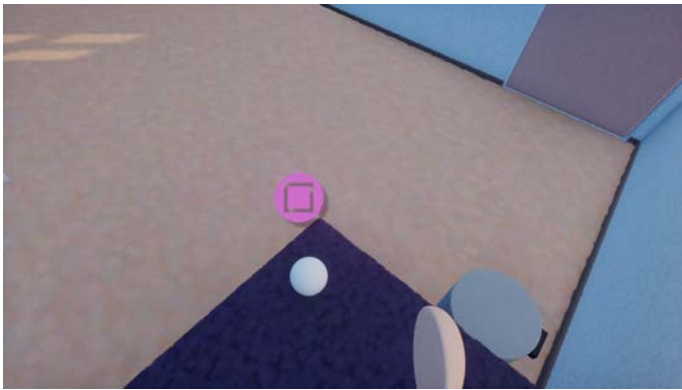
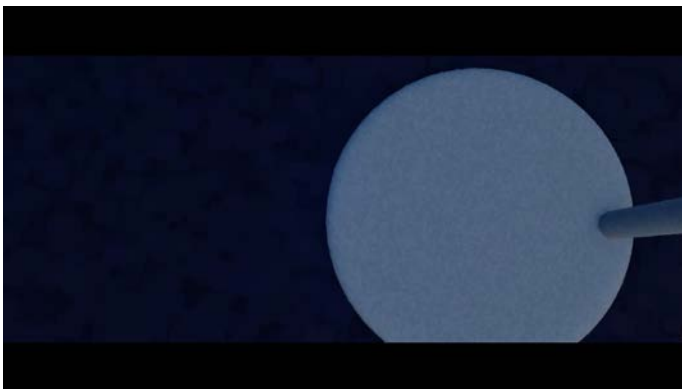
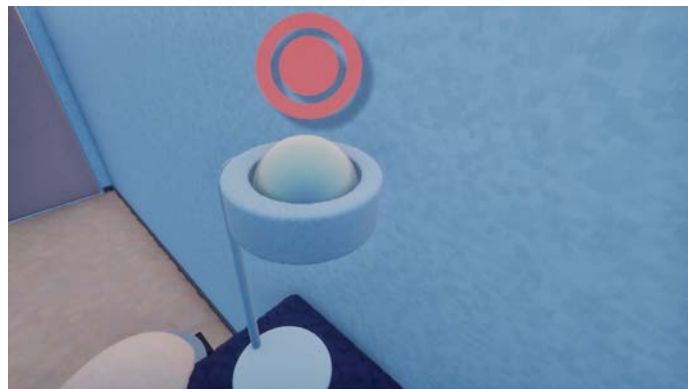
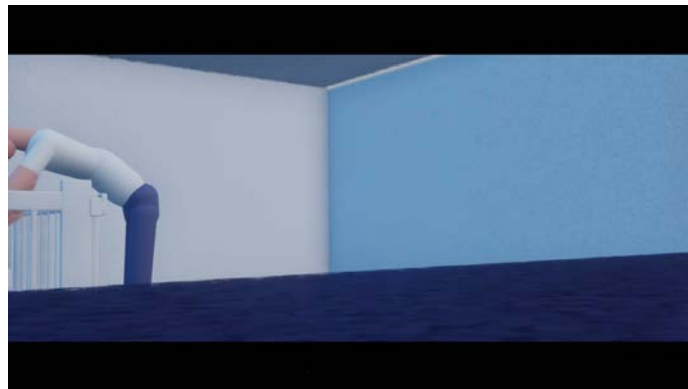
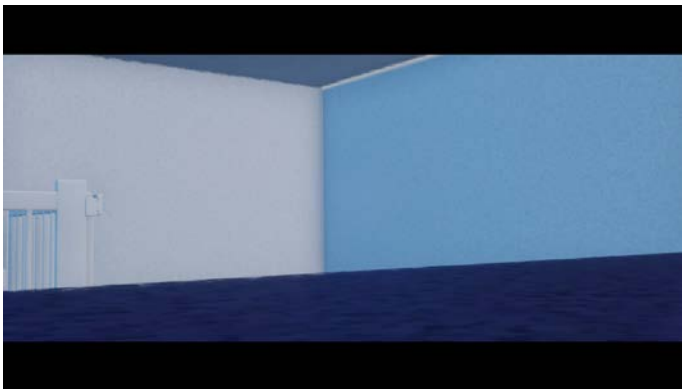


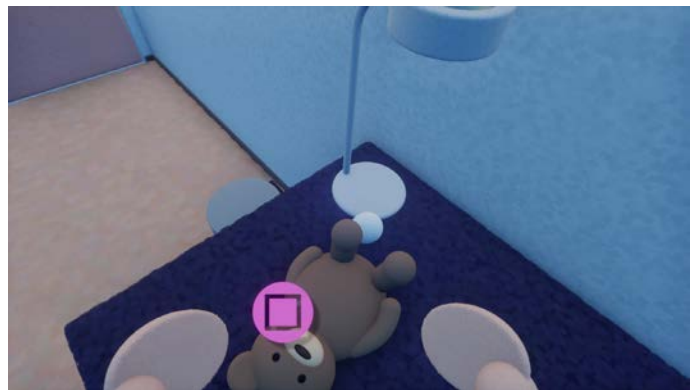
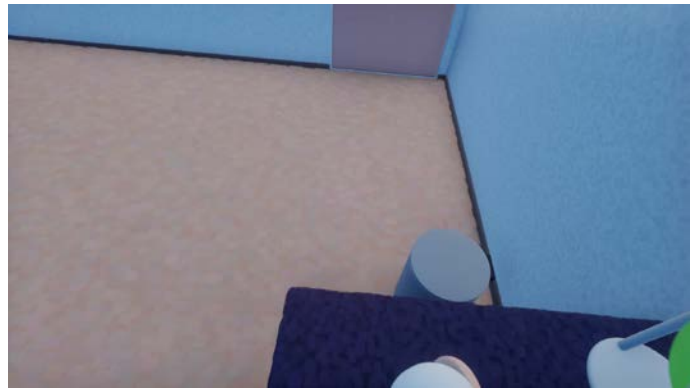
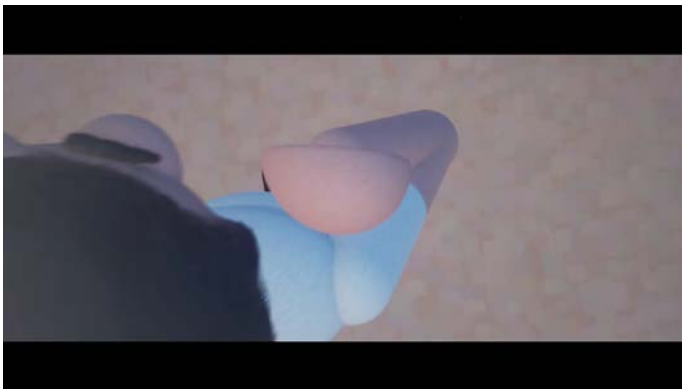










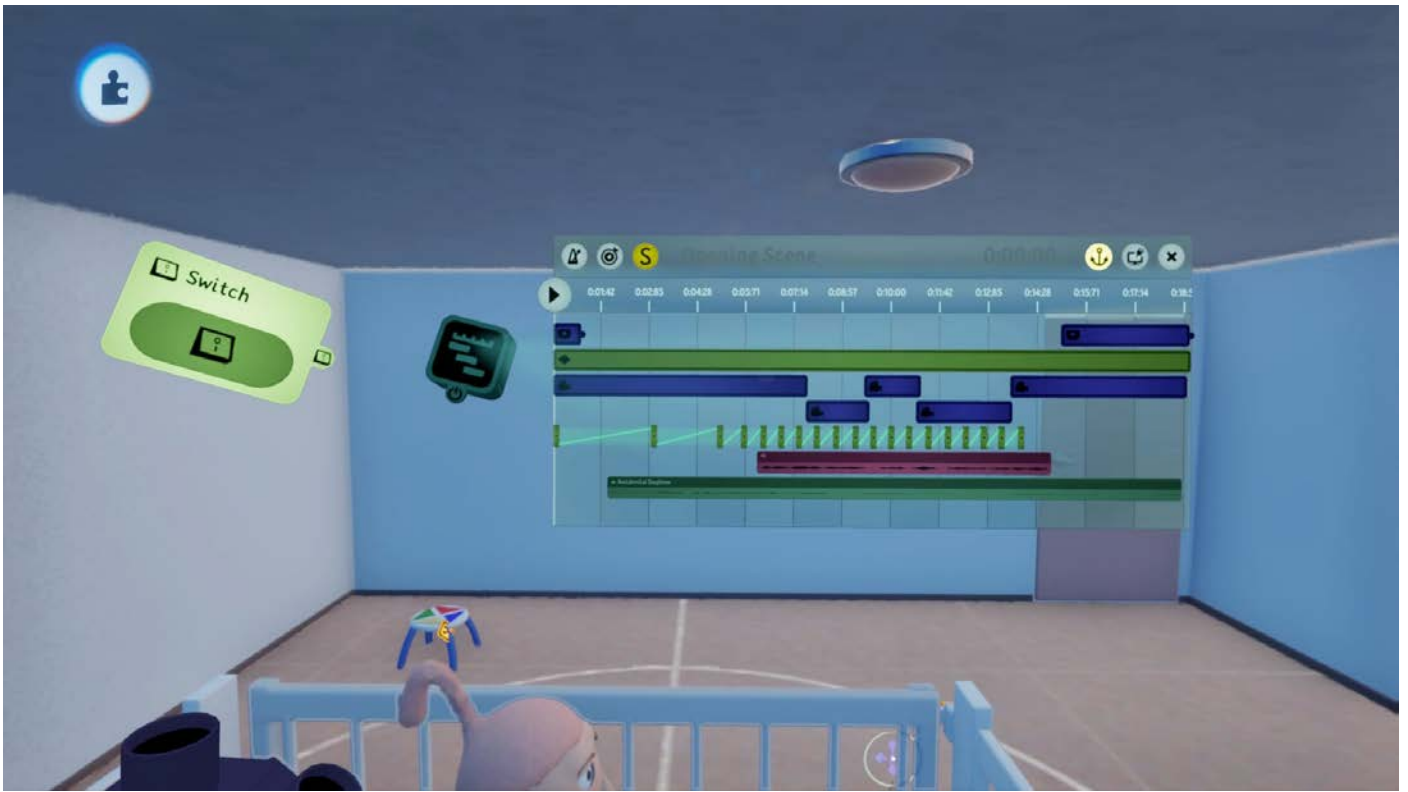






demo |

***FINAL LOGIC***



Opening Scene



Latch



## Pillow



## Blanket





## Barrier



## Middle Scene





## Activity Table



## Trashcan



**Stool**



**Climbing Stool**





**Climbing Drawer**



**Baby Monitor**



## Light



## Ball





**Stuffed Animal**



**Player Puppet**



demo II

# ***PREDEVELOPMENT***

## GOAL

I had conceptualized my first goal and created a demo accordingly. I decided to reuse my assets and make another demo with a different goal. I still wanted to have the player interact with objects in an alternative way, but I needed to figure out in what way.

I came up with a few ideas. The first one was where the baby would need to find her way back to the crib. The objective was in line with the previous demo I came up with, but I had already shown the player the functions of the objects, so it did not seem that it was offering a different type of perspective.

The next idea I had was for the player to play as the bear and try to get to the baby, but that would basically be the exact same interaction as the baby trying to get back to the crib. I thought it might be interesting to make some type of upside-down world, but I could not conceptualize a way to make that work. I needed to come up with something a little more realistic.

The last idea I had was to have the baby's room be a mess and the player would have to clean it up. The player would have to put all the objects back to specific spots. The player would know where these spots were based on the interactions he/she would have with the objects. I thought this idea would be a good modification of my first level and was doable. I also chose to add a timer to the level, so the game would restart if the player did not finish the level quick enough.

## OBJECTS

I had already created the objects, but I needed to rethink their functions. Since the goal was to put the objects back to their specific locations, I thought about which objects to use. I wanted the actions to be different in this demo, so I did not want the player to climb up the changing table again, so I left the baby monitor and desk lamp alone. I did, however, decide to use the ball and stuffed animal, but they would be placed somewhere else.

I wanted all the objects that the player would be interacting with to be scattered over the floor. These objects would include: the activity table, trash in the trashcan, blanket, pillow, stool, stuffed animal, and ball. Like before, I created a flowchart to better visualize the actions (see figure 36). Apart from the ball and stuffed animal, I used the objects' locations from the first demo as the locations they should be moved to in this demo.

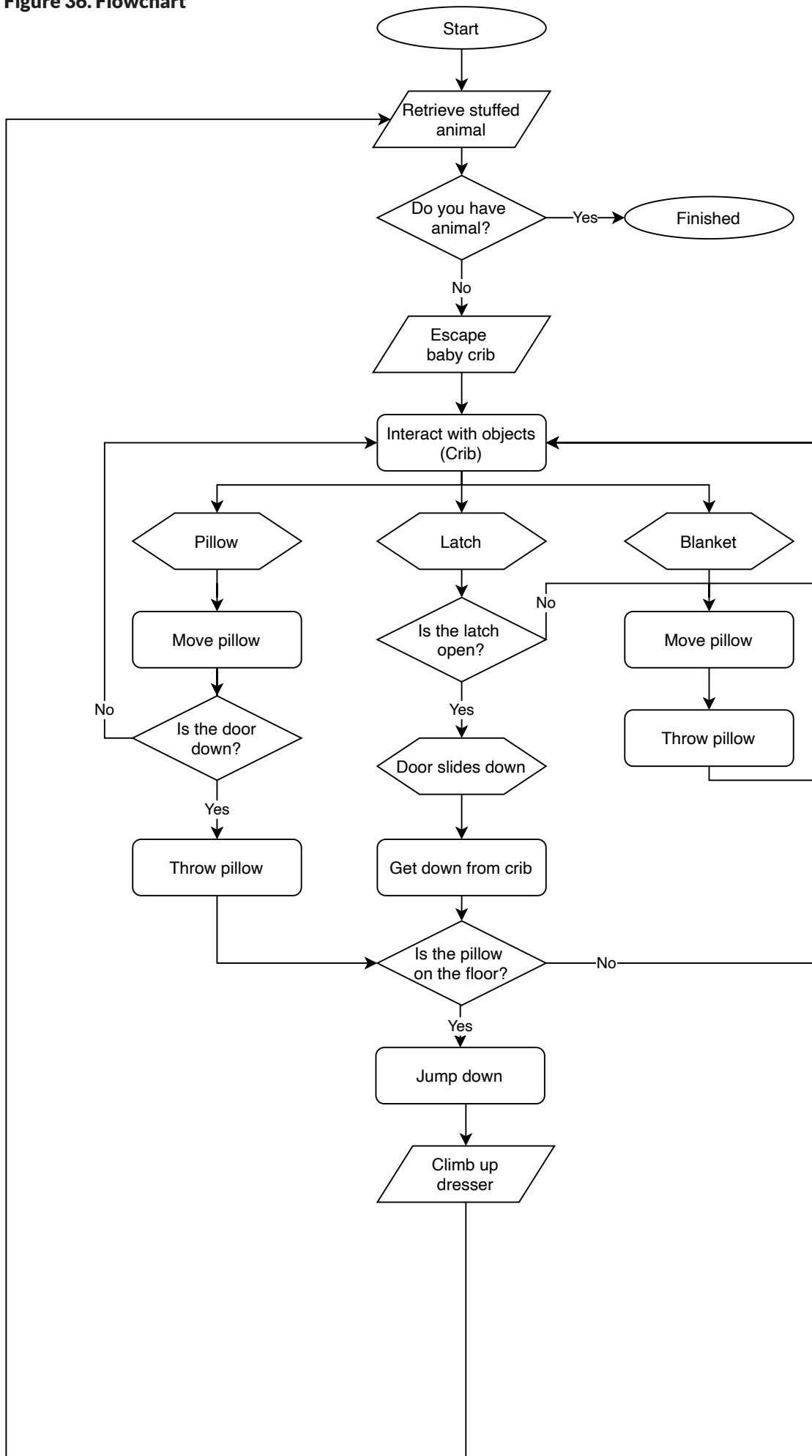
## CUTSCENE CONCEPTUALIZATION

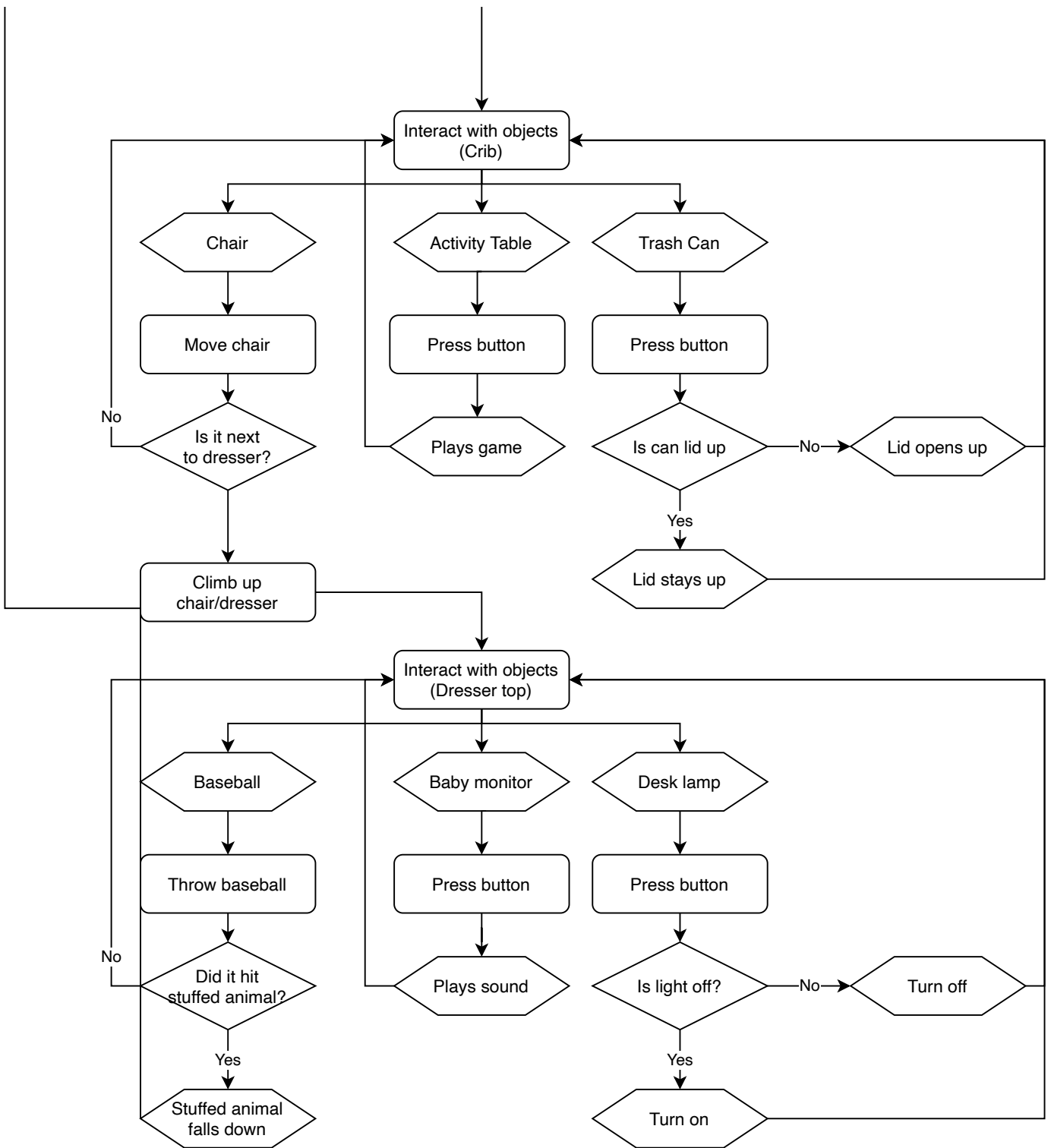
The objects needed to show the player where their proper locations were in the cutscenes, but I needed to figure out how. Before I had made the cutscenes so they were scenes being passively watched, but I wanted to do something different this time. I decided the player would be able to control the camera during the cutscene and use that free-look to determine where the object is located.

I would use this for all of the objects, apart from the stuffed animal. In the stuffed animal's cutscene, it would be on the shelf like in the previous demo, but it would zoom in on the crib. This action would signify that the stuffed animal should be placed in the crib and was a callback to the first demo, where the objective was to have the stuffed animal in the bed with the baby. Here is the storyboard I created to show the progression (see figure 37).

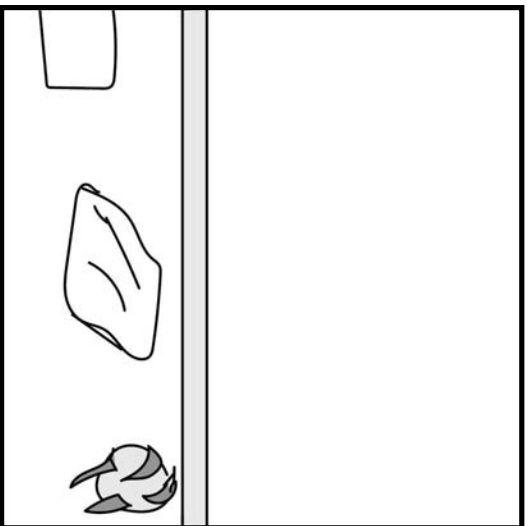
The level needed an opening and closing cutscene, as well as one if the timer ran out. I chose to do a black screen with white text over it. The opening card would say, "You need to clean up and go to bed!", the ending card would say, "Good on you, you cleaned your room and made it to bed before pops came!", and if the timer ran out it would say, "Oh no, you took too long, loser!" With predevelopment done, it was time to move to development.

Figure 36. Flowchart

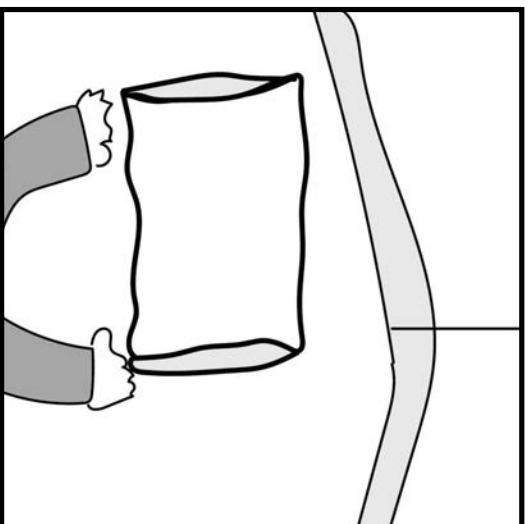




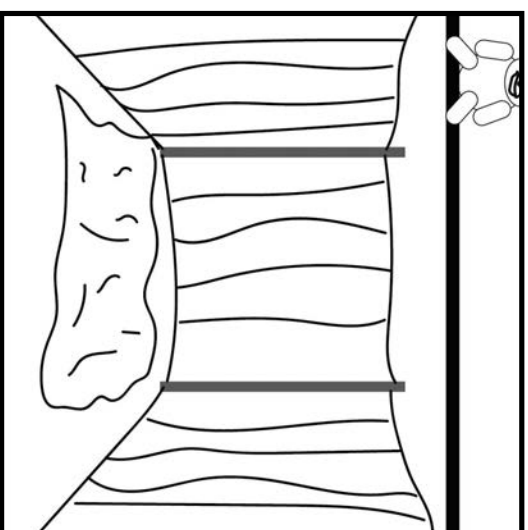
**Goal: Put all objects back to specific locations**



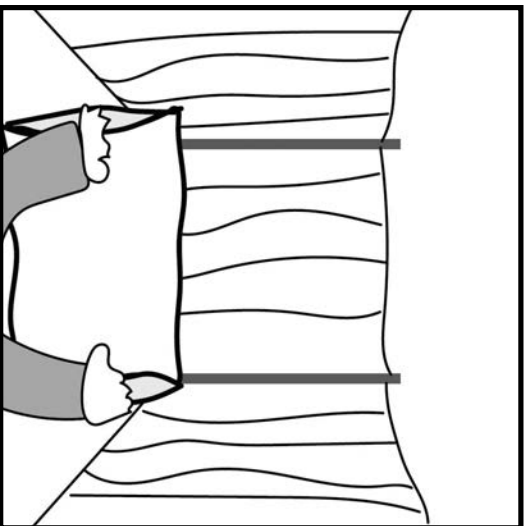
Action: Baby is on floor in room and needs to start interacting with objects.



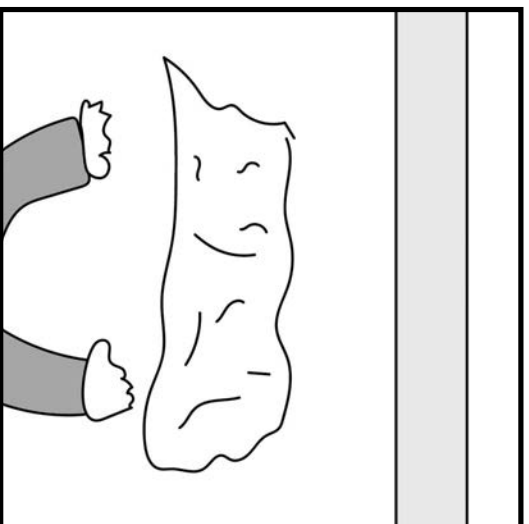
Action: Baby interacts with pillow.



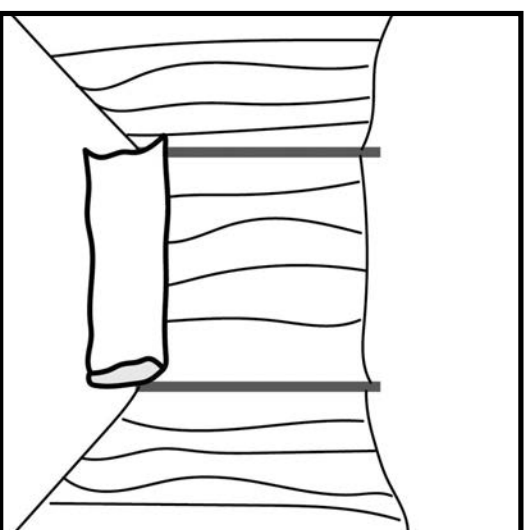
Action: Camera goes into view of pillow in correct position.



Action: Baby puts pillow where the pillow point of view (POV) was.



Action: Baby interacts with blanket.

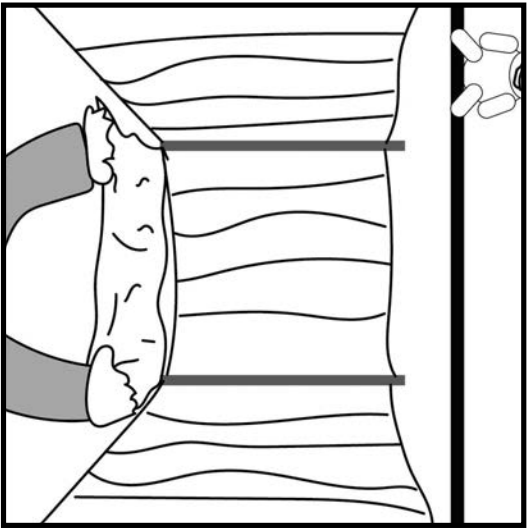


Action: Camera goes into view of blanket in correct position.

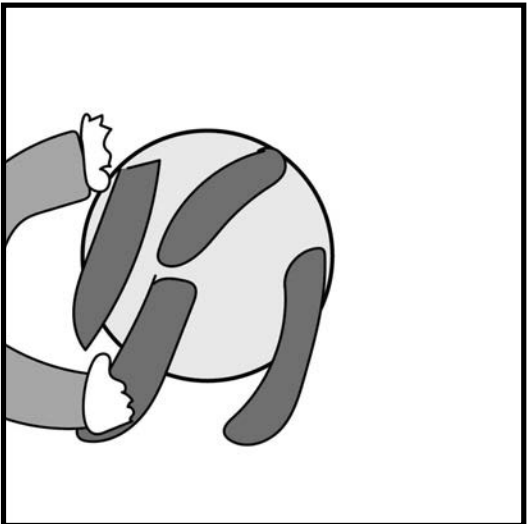
Figure 37a. Storyboard



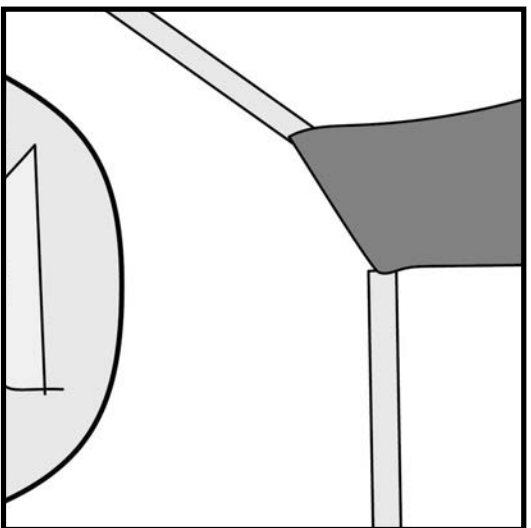
Figure 37b. Storyboard



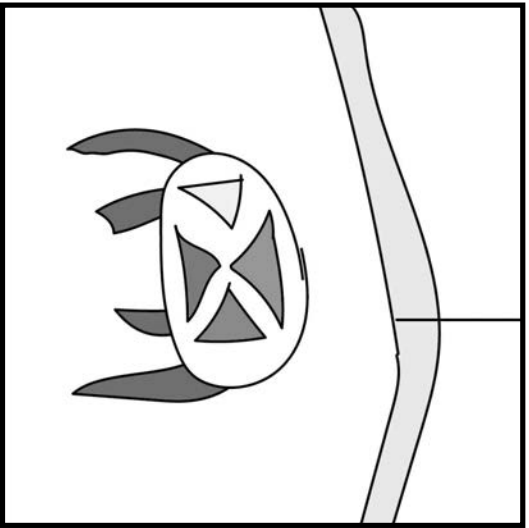
Action: Baby puts blanket where the blanket POV was.



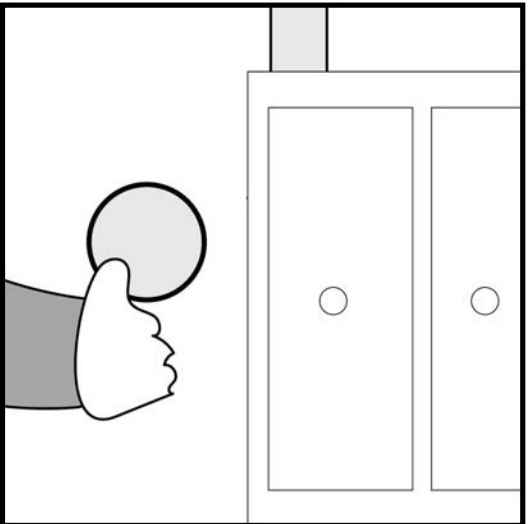
Action: Baby interacts with activity table.



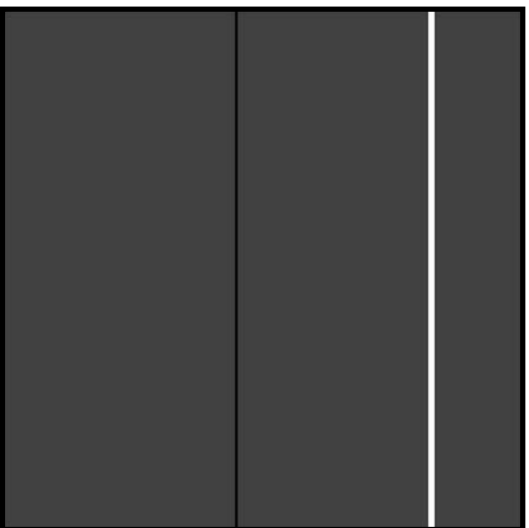
Action: Camera goes into view of activity table in correct position.



Action: Baby puts activity table where the activity table POV was.

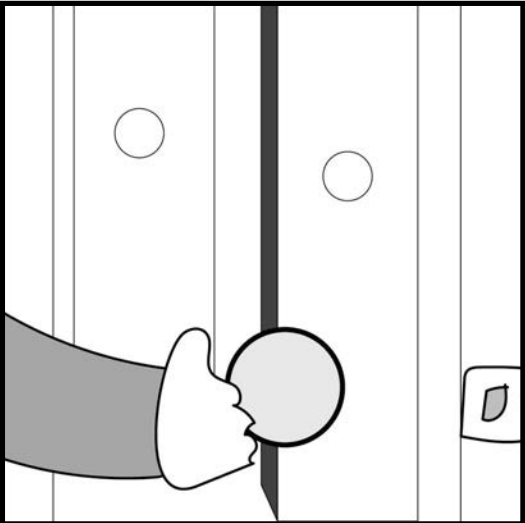


Action: Baby interacts with ball.

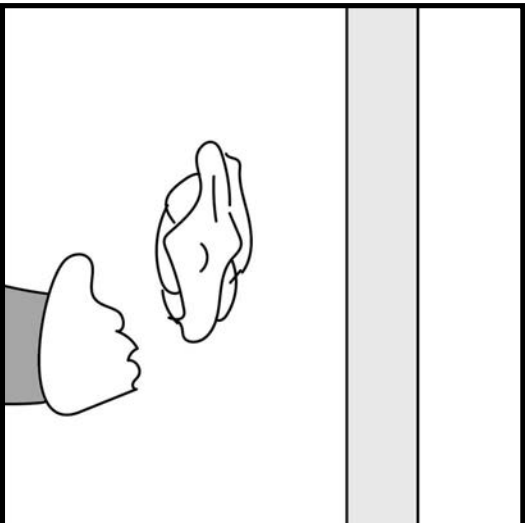


Action: Camera goes into view of ball in correct position.

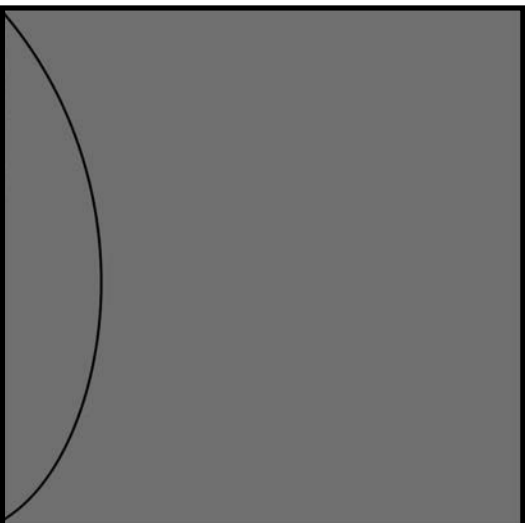
Figure 37c. Storyboard



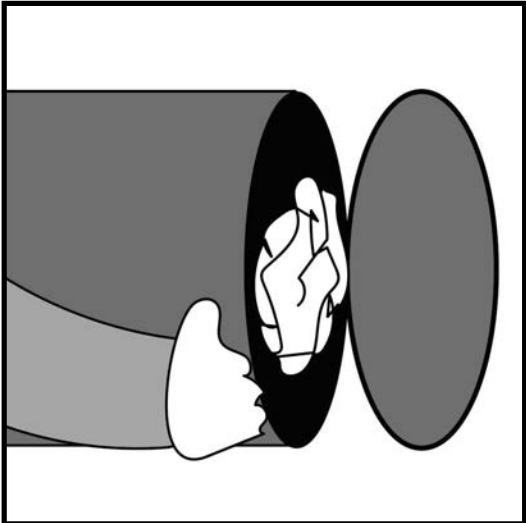
Action: Baby puts ball where the ball POV was.



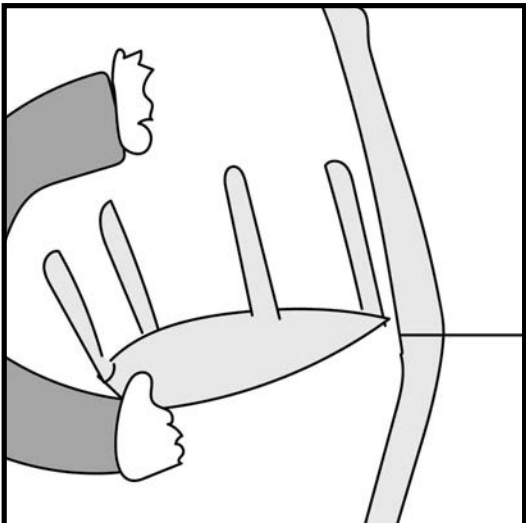
Action: Baby interacts with trash.



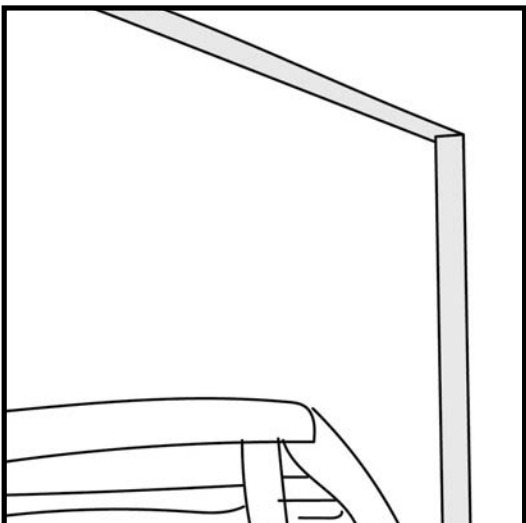
Action: Camera goes into view of trash in correct position.



Action: Baby puts trash where the trash POV was.

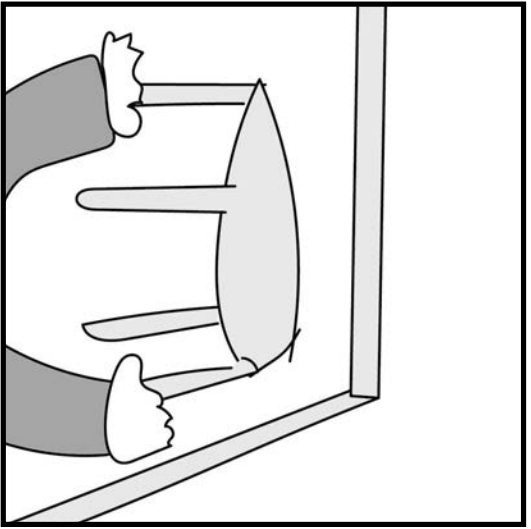


Action: Baby interacts with stool.

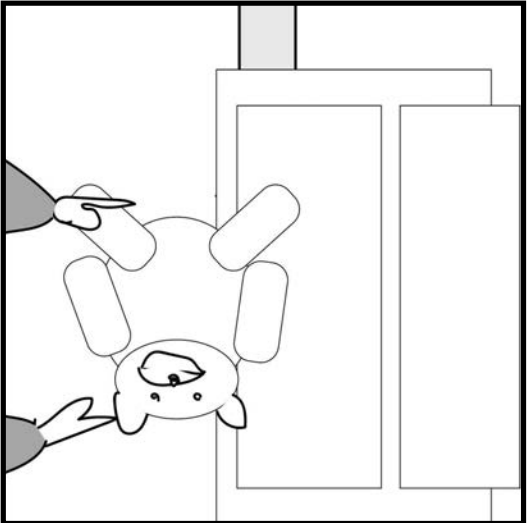


Action: Camera goes into view of stool in correct position.

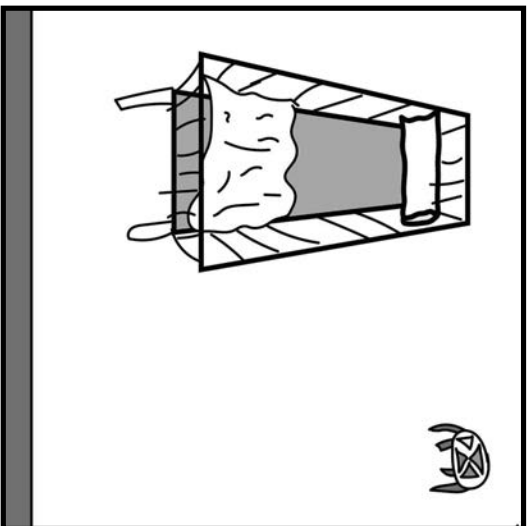
Figure 37d. Storyboard



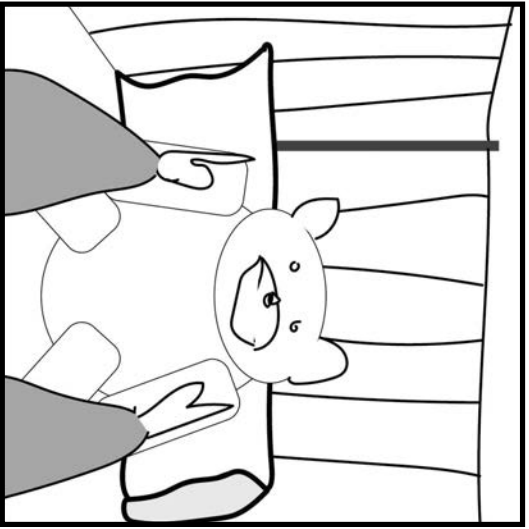
Action: Baby puts stool where the stool POV was.



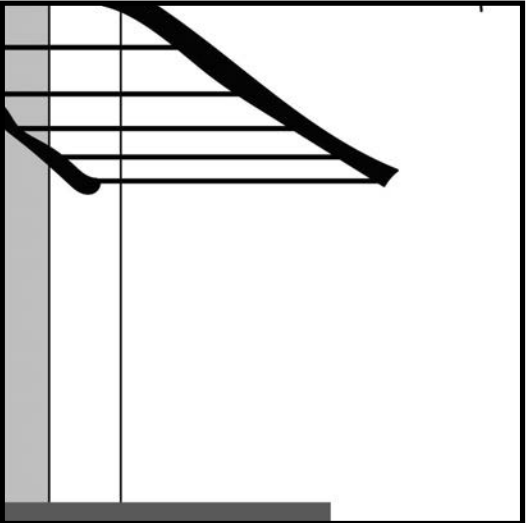
Action: Baby interacts with stuffed animal.



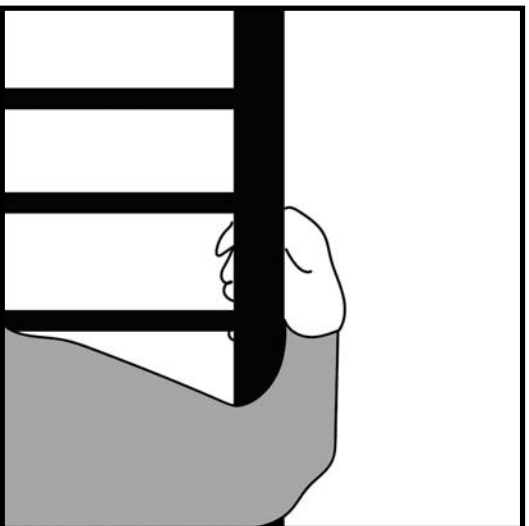
Action: Camera goes into view of stuffed animal and zooms to where the stuffed animal wants to be.



Action: Baby puts stuffed animal where the stuffed animal POV was.



Action: Baby closes crib door.



Action: Baby closes the latch.

demo II

***DEVELOPMENT***



# THE ACTIVITY TABLE

To begin the development, I made a copy of my first demo. I would use the copy as the base for this level. The first bit of logic I wanted to work on was making the first person perspective free-look for the cutscenes. I arbitrarily chose to begin with the activity table. There was already the logic in it to press circle to start the timeline, so I deleted the logic for everything after the timeline as well as everything that was in the timeline.

In my first attempt to create the free-look, I copied the two cubes and their logic from the rig I made for the playable character. I scoped into the activity table. I moved the camera gadget from the microchip in the top cube into the timeline. I adjusted the camera so it was still in the proper location on the top cube.

I connected the right stick from the activities table chip to the chips in the cube and made sure that the controller sensor had the same settings as the player's one. The important part was to force possession. I made a tag in the activity table called, "Table", and selected it for the teleporter in the bottom cube. I made a keyframe in the table's timeline, where I turned off the playable character, which extended for the entire timeline.

My logic worked in the sense that the view switched from one rig to the other one, but I still had a problem. Although the perspectives would change, the navigation in the activities table rig was not acting properly. Pressing right would go up, and up would go left and down, etc. I needed to figure out what was causing that problem. I troubleshooted the rig, the top box would rotate properly on the Y-axis when I manually moved it and the rotator for the X-axis was set properly.

While I was trying to figure out a way to fix that, I thought why don't I just use the same rig that I use for the playable character. It already had the settings I needed and it was calibrated, I only needed the rig to be teleported to the activity table instead of the character. I accomplished this by copying the

"Player" tag into the activity table and attaching a switch to it to power it on and off. I attached a switch to the "Player" tag in the player puppet, as well. I made a keyframe for the entire timeline that turned off the "Player" tag in the player and turned on the "Player" tag in the activity table.

What I did worked, but I needed to make a few adjustments. The camera placement was not where I would have liked it on the activity table, so I adjusted the position of its "Player" tag to make it so the camera's view was right above the table (see figure 38). In the keyframe, I made the player puppet non-visible, as opposed to turning it off because I still needed the controller sensor to work to move the camera rig around.



**Figure 38. Activity table camera rig logic**

Now that I had the important logic figured out for the timeline, I added more actions to the keyframe. All the objects were in their desired locations; I animated them so that when I move them later they would still be in their proper positions in the timeline. I just turned the ball and trash off because the player would not see them when they are put away. I also closed the crib and hatch, and the top right drawer in the changing station. The crib door and latch would be open in the beginning of the level.

I decided that a good time for the timeline would be seven seconds. I added a wiper to the beginning and the end of the timeline because I wanted a little

bit of blackness while the camera switches from the player to the object and vice-versa. I made them about half a second long. I had the beginning of the first and end of the second aligned with the keyframe, but there was issues with the camera rig being teleported at the end, so I made the keyframe around a quarter of a second shorter to give it more time during the end transition (see figure 39).



**Figure 39. Activity table timeline**

The timeline was finished, so I moved on to the rest of the logic for the activity table. First, I had the timeline send a signal to a counter once it had played, and had that send a signal to an AND gate. I had square from the controller sensor and the trigger zone go into an AND gate, then I had this AND gate export to the AND gate that had the signal from the counter and they sent their signal into another counter. For this level, I always wanted circle to activate the timeline, square to pickup, and triangle to drop/throw.

I already knew the pickup logic, so I had the last counter output to the power of another controller sensor and a teleporter set to the puppet's "Chest". I already had a gesture keyframe connect to the trigger zone, and I copied the stool holding gesture and had the last counter output to that as well. I took the pick up and drop sounds from the stool and connected one to the last counter and one to when the player pressed triangle.

I noticed when I picked up the activity table, the lights on it would stay in place and not move with the table. This was because the chips was snapped on the table and not on the level the table and lights were grouped. I grabbed the chip, scoped out to the correct level and snapped it back on the table. I also made the grouped object movable. Those actions solved the problem.

The last thing I needed to do for the chip was figure out the text displayer. I wanted it to first show circle and then afterwards display square, and I wanted to display triangle when the player picked it up. For the circle displayer, I had a signal sent from the trigger zone into it. It was powered on by the A output of a selector and once the player pressed circle, the first AND gate would change the selector to B.

For the square displayer, the trigger zone and the counter after the timeline go into an AND gate that would play it. It was powered the same way as the circle displayer. The output from the last counter would cycle the selector to the next output and pressing triangle would cycle it to the previous. The triangle displayer was simple, just outputted a signal from the last counter to the displayer. As I did with all the text displayers, I adjusted the size and position, and changed the button displayed if it was needed.

It was time for me to move the activity table to its messy position. I wanted to lay it on its side, but the distribution of its weight would cause it to flip onto its top. To fix this I sculpted a box next to it, so it would lean against the box and not fall over. I made the box invisible, not collidable with anything, and changed its label to foe. In the activity table, I changed the settings, so it collided with foes.

This held the table up, but I only needed the box there for before the player picks it up. I snapped a destroyer onto it to get rid of it. I had the AND gate in the table that went into the last counter output to power on the destroyer (see figure 40).

I thought I had finished everything for the activity table, but I had missed one thing, I wanted the layer to be able to continuously play the timeline,



**Figure 40. Activity table destroy logic**

so he/she could recheck the object's desired location. Before, I had the first AND gate go into a counter and then into the timeline, that made it so the timeline would only play once. I changed the playback mode of the timeline to once, but I deleted the counter so every time the player would be in the zone and press circle it would send a signal for the timeline to play (see figure 41 for logic).



**Figure 41. Activity table logic**

I decided it would be good at this point to delete the assets and logic I did not need. I deleted the dad puppet and the extra baby puppet. I got rid of the of the barricade for the crib and the one under the

open drawer, as well as the little ramp in the drawer. I also deleted the logic on the lamp, baby monitor, and changing station.

I got rid of the microchip that zoomed in on the stuffed animal from the floor and the logic for the opening cutscene. I deleted all the cloned objects. I only wanted the player to be able to throw the ball and the trash, so I deleted the emitters for the pillow and blanket that were in the puppet's microchip.

## THE BALL

The next object I worked on was the ball. I basically already had all the logic I needed for the ball, I just needed to have circle start the timeline, change the text displayers, and add the "player" tag, like with the activity table. Once I did that, I went into the timeline and deleted everything. I copied the wipers and the keyframes into it from the activity table. I went into the keyframe and made the ball visible and I moved it inside of the closed drawer, and I changed it so the "Player" tag in the ball would turn on.

Since I wanted the player to be able to play the timeline whenever he/she pressed play, I could just clone the ball. I set the clone as the object to be emitted in the puppet's microchip and only connected the cloned ball to it because the original ball was still connected to it. I deleted the circle text displayer and selector that went into it on the clone because the player only needed to see circle the first time when they came to the ball. I moved the original ball to a position on the ground near the changing station.

When I was testing it, I noticed that the camera would have issues in the timeline of the cloned ball. I fixed this by creating a third ball that I put in the shelf. I snapped a microchip on it. Before in the balls, I had the trigger zone and circle on the controller sensor output to an AND gate which outputted to the timeline. I moved an AND gate and timeline into the third ball and connected the other two balls triggers and sensors to that AND gate and then the



timeline outputted to the counter in the original ball. The AND gate also outputted to input B on the original ball.

The reason the third ball did not output anything to the second ball is because it did not need to output anything. The clone did not need a circle display. All the second ball needed was the two wires that connected the third ball, and the trigger and square button output that went into the AND gate, which activated the counter to pick up an object.

I deleted the “Player” tags in the first two balls and put one in the third ball. I had that tag turn on in the keyframe and I adjusted its position, so the camera was right above the ball in the cutscene. I made the ball invisible and non-collidable, but visible in the keyframe. See figure 42 for ball logic.

Since I wanted the drawer to be shut, I needed to add that logic, as well, for when the player threw the ball in there. The logic was the exact same as that to open the trashcan, so I copied that microchip and copied it onto the drawer. I changed the text displayer text from circle to cross.

I modified the gesture keyframe, so the palms in the hand were facing forward and the fingers up. I de-animated the trashcan open keyframe and made

it so the drawer was in the changing station, setting the slower power down and up to one second. I also changed the closing and opening noise to the cupboard roll loop.

After I created the drawer action, I noticed that the cross displayer would be visible in the timeline. I fixed this by turning the drawer’s chip off during the keyframe. Later, I saw that there was similar issues with the ball, depending on where it and the puppet were located. I solved this the same way as before, by turning off the microchips for the first two balls during the timeline.

The drawer was pretty barren, so I decided to fill it again with a pink box that would take up most of the space. I used the impressionist fleck and set the looseness to around sixty percent. This made the block look like a folded blanket. I left enough space so the ball could still be thrown in there. I went back into the closed keyframe for the drawer and moved the box to where the drawer moved.

I did not want the ball to get stuck on top of the changing station because there would be not way to get it. I created a large box that I placed on top of the changing station to act as a barricade. I made the box non-visible.



Figure 42. Ball logic



## THE TRASHCAN

The next object I would work on was the trash. I first changed the trashcan to display cross and open with it. I changed the ease-in and -out of the open keyframe to half a second. I cloned the trash in the can and moved it in front of the crib; this would be the one with which the player would first interact.

The logic for the trash was the same as the ball since I wanted it to be thrown into the can. I copied the logic chip from the original ball to the trash in front of the crib. I copied the trash, then snapped a copy of the second ball chip on that. I adjusted the transporter location on both of them, so that it would be in the correct position in the player's hand.

I copied the microchip from the third ball onto the trash in the can. In the keyframe, I changed the "Player" tag off in the ball and on in the trash; I also adjust the tag's position. I connected the other pieces of trash to the one in the trashcan, like with the balls. I went into the player's chip and copied the emitter and I renamed it trash. I selected the proper piece of trash to be emitted and changed the emit speed to 2.0 m/s. I connected the trash outside of the can to the emitter.

I was having issues with the trash falling beneath the can when thrown in there, so I made the emitted trash movable, changed the weight from 0 to 1 percent and made the physical cost high. This did not help, so I labeled the can scenery and had the trash collide with that label. I also made the friction ten percent.

## THE PILLOW

I moved onto the pillow. I copied the logic from the activity table to the pillow, but I kept original sounds and arm gestures, and I moved the pillow onto the floor by the open crib door. I adjusted the position of the teleport and "Player" tag. I went into the keyframe and turned the pillow's "Player" tag on and turned off the ball/emitted ball and trash/emitted

trash. I did that for the keyframe in the activity table's timeline, too.

When I played the timeline, I was having clipping issues with the camera. I made the different parts of the crib non-collidable, but that did not work. I fixed it by making the pillow non-movable in the keyframe.

The next problem I ran into was that the crib door, depending on where the player was standing, would collide with him/her after the timeline played. I wanted the player to not collide with the door for the duration of the timeline, so I had the AND gate that went to the timeline also go to a counter set to one. I had the counter output to a keyframe, where I made the door non-collidable with friends. I did not want this effect to last forever, so I had the counter also output to a timer set to twelve seconds. After the twelve seconds the timer would reset the counter, the timeline was only around seven second, so that was enough time (see figure 43).



Figure 43. Pillow logic

## THE BLANKET

The blanket was the exact logic as the pillow, apart from the extra logic for the door. I adjusted the transporter and "Player" tag location. I went into the timeline, and did the usual changes. I made sure to also make the blanket non-movable in the timeline, as well, because it was having similar issues as the

pillow in the timeline. I moved the blanket onto the ground on the right of the blanket.

## THE STUFFED ANIMAL

Next object I worked on was the stuffed animal. I deleted all the logic in it, apart from the hand gesture. I copied the logic from the blanket into its microchip, apart from the “Player” tag and switch. Since I was not going to use free-look in this timeline, I did not need it. In the timeline, I de-animated the “Player” tag in the blanket and the player, but I left the player invisible.

I added two camera gadgets into the timeline. I placed the first camera so that it was the view of the stuffed animal on the shelf. I changed the transition to cut and the transition time to zero. I made the gadget a little longer than a second.

The second camera I made the rest of the duration of the timeline and it ended when the keyframe did. I moved it in closer to the crib, so it would look like the camera was panning in. I kept the smooth transition and changed the transition time to five seconds. With that I moved the stuffed animal to the ground, bottom right of the blanket. See figure 44 for stuffed animal logic.

## CLIMBING THE CRIB

I needed to add logic to the crib, so the player could climb up to place the blanket, pillow and stuffed animal. I had the previous logic in the player for climbing up with the teleporter and noise, I just needed the logic for the crib. I snapped a microchip onto the mattress of the bed. In it, I put a tag labeled “Bed”, a trigger zone, a text displayer, and a hand gesture keyframe.

The gesture was arms out, up, and palms down like the baby was going to lift herself up. I made the trigger zone a rectangular prism around the open side of the crib and connected it to the text displayer, and to the AND gate in the puppet that would activate the transporter when cross was also pressed. I adjusted the position and size of the text displayer, and set it to display cross.

I had the AND gate also output to a “Wooden Splintering” noise. I turned the volume down to thirty percent and went into the slice editor and tweaked the sound to the snippet I wanted. I deleted the other sound files in the editor. See figure 45 for logic.



Figure 44. Stuffed animal logic

## THE CRIB DOOR

Once all the objects were in place, the player could close the crib door, when the door was closed, the player could close the latch, then level would be completed. I started to make the logic for the door. I snapped a microchip onto it. I put a controller sensor and trigger zone in it. I set the zone as a cube around the part of the bed next to the door and detect the “Player”. I had the trigger, controller cross output, and AND gate from the “Objects Locations” output in an AND gate.

The AND gate would output to counter, set to one, and the counter would output to a keyframe of the door being closed. I set the slow power up/down of the keyframe to three seconds. The counter also outputted to a “Tiny Hinge Squeak Loop” sound. I turned the volume down to forty percent and the playback to once.

I had the trigger zone output to a keyframe with a hand gesture with the left arm out, palm forward, and fingers up, with same settings as the other gestures. I needed to display the “action” button, so I had the trigger zone and the AND gate from “Object Locations” output to an AND gate, which outputted to the text displayer. See figure 47 for logic.



Figure 47. Crib door logic



Figure 45. Climbing crib logic

## OBJECT LOCATIONS

The logic in all the objects was done and the player could get to all the locations; the next step was to add trigger zones for the objects to tell when they are in the right locations. I made a tag in each object, so they could be detected. I made seven tags: one for the stuffed animal, blanket, pillow, stool, activity table, ball and trash. For the ball and trash, I only needed the tag in the emitted objects.

I made a microchip and stamped it in the middle top of the room. I labeled it, “Object Locations”. Inside it, I stamped seven trigger zones. I set each one to detect each of the objects, and I set all the zones in the locations where the objects are in the timelines, apart from the stuffed animal, whose zone I placed in the crib. I used the cube shape for all the objects apart from the trash, where I used a cylinder to fit the trashcan. All the trigger zones went into an AND gate (see figure 46).



Figure 46. Object location logic



## THE LATCH

The last item was the latch. I used the same logic as the crib door to close it and for the text displayer, the only difference is that instead of using output from the AND gate in “Object Locations”, I used the output from the counter in the door’s logic. I changed the gesture, so the palm was facing the character and the fingers were down. I made the keyframe after the counter of the latch closing, with a slow power up/down of two seconds. I did not want the text displayer to show after the latch was closed, so I had a selector output A to power the text displayer and the AND gate, that closed the latch, output to Input B of the selector.

I wanted the demo to end here, so I had to added a little bit of more logic. I had the AND gate, that closed the latch, output to a timer, as well. I set the time to 1.5 seconds and had it output to a timeline. I set the timeline to ten seconds. I dropped a wiper and text displayer in it that extended the whole time. At the end I had my doorway, set to exit, which ended the level.

I had the wiper make the whole screen black and had the text displayer show, “Good on you, you cleaned your room and made it to bed before pops came!” I changed horizontal alignment and horizontal alignment (screen) to center, and vertical alignment and vertical alignment (screen) to middle. I enlarged the text until it took up around twenty percent of the screen.

When I would activate the timeline it would only play for a second. I changed the fade-out for the wiper to ten seconds, but it did not fix it. I got rid of the doorway, it did not work, and I took the displayer out of the timeline. I realized that the problem was the timer, since the timer only pulses, I needed a counter after to hold the signal, so I put a counter between the timer and timeline and that solved the problem. See figure 48 for logic.



Figure 48. Latch logic

## OTHER LOGIC / SETTINGS

I had the ending of the demo finished, but I still needed the opening. I copied the ending timeline and stamped it into the scene above the crib. I shortened it down to seven seconds and deleted the doorway. I changed the text to, “You need to clean up and get to bed!” I increased the fade-out to 1 second. I added a switch to power on the timeline, so I could turn it off while I was testing the level.

I wanted to change the lighting in the room to make it darker, so I adjusted the sun and sky properties. I changed the sun brightness to fifteen percent and the sky to thirty percent. This made it darker, similar to the late-afternoon, early evening.

I decided to add some background music to the scene, so I searched the Dreamiverse for something that was tranquil. I found “Zen Garden (Generation)” and it matched well with the mood I wanted. I stamped it into the scene left of the “Object Locations” chip. With the chip stamped in the scene it would play the song on repeat. I turned the volume down to -20% because I did not want it to be too loud.



The next item up was the level timer. I stamped a microchip into the scene right of the “Object Locations” chip. I snapped a timer inside set to 480 seconds (8 minutes) and set it to countdown. The timer pulsed to a counter which outputted to a timeline.

I copied the timeline from the opening timeline, changed the duration to five seconds, fade-out to 0.5 seconds, and the text to, “Oh no, you took too long, loser!” After the timeline played, it outputted to the reset scene input in a global setting chip. I added a switch to turn on the timer for when I wanted to test the demo without it. See figure 49 for logic.



**Figure 49. Death timer logic**

demo II

## ***IMPROVEMENTS***

## TESTING / FIXING BUGS

Through changing little things here and there, I noticed that sometimes the different timelines in the objects would have issues, where the objects would not be in the right place when the timeline would play. I needed to make sure that all the objects were in the correct positions. I went through all the timelines and moved any object out of place into the right position.

I talked earlier about how I turned off some of the logic in objects during timelines. I went back into each objects' timeline and turned off the other microchips when the timeline played. I did this just to make sure there would not be text displayers being shown during any of the timelines.

Like with my first demo, I had many issues with the crib door and latch. There was the problem I already mentioned that I solved by making the objects non-movable in the timeline, but then there were problems with the latch jiggling in the timelines or remaining closed after a timeline played. The first thing I did was make a separate keyframe for the door and the latch in each timeline. I made the keyframe for the latch a little smaller than the door, so it would open and close before the gate would move. This helped, but did not solve the problem.

I decided on changing the physical cost and friction of the latch. I switched the cost to high and the friction to zero. This did not work. I changed the friction to ten percent and put its weight down to ten percent. That fixed the problem.

The last problem I had was with the trash, the problem still persisted where it would disappear into the can after the player threw it in there. When that happened, the trigger zone would not pick it up and the player could not finish the level. The solution I used for this was to put a counter after the trigger zone for the trash, so once it hit that zone it would keep sending a signal to the AND gate in "Object Locations".

## POLISHING LOGIC

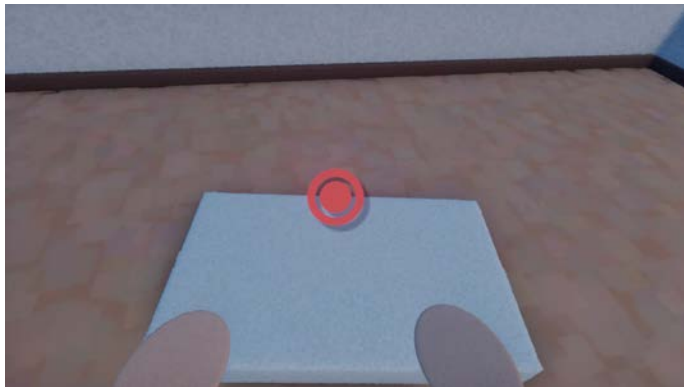
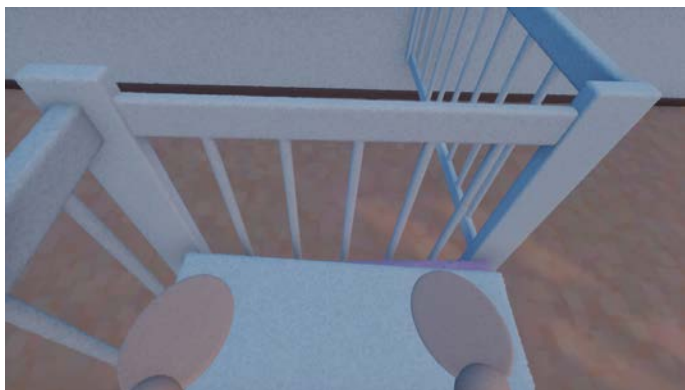
I did need to do much to clean up the logic. I had the power switch for the "Player" chips in each object, which I would turn on during their timelines, but I realized later that the switch was unnecessary. I deleted all the switches and turned off the power on the tags themselves. I went into each timeline and just powered on the tag for the appropriate object.

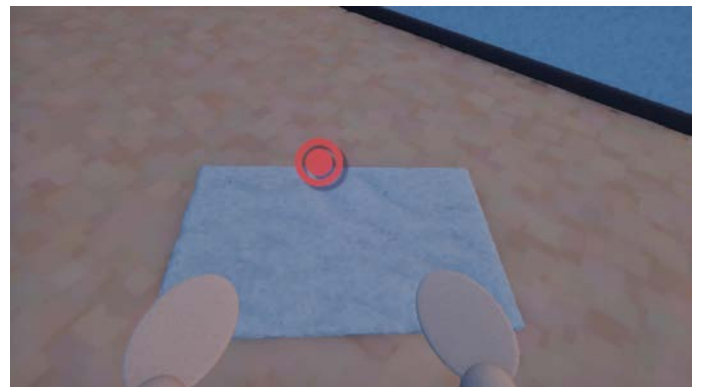
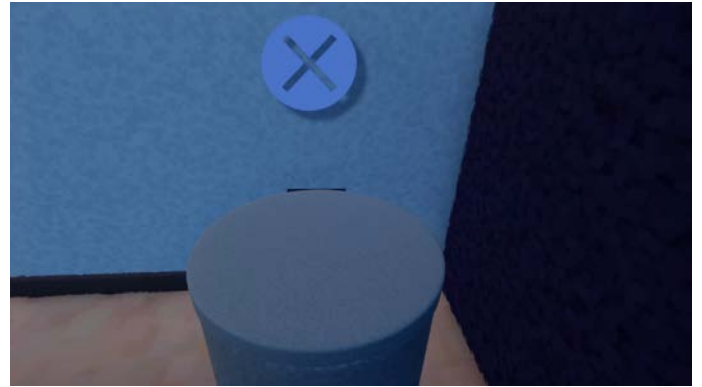
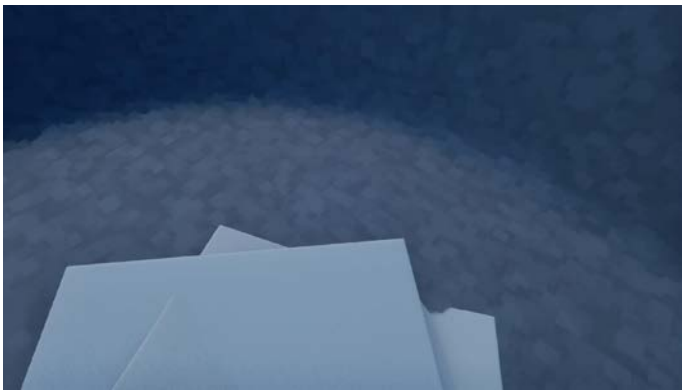
demo II

***PLAYTHROUGH***

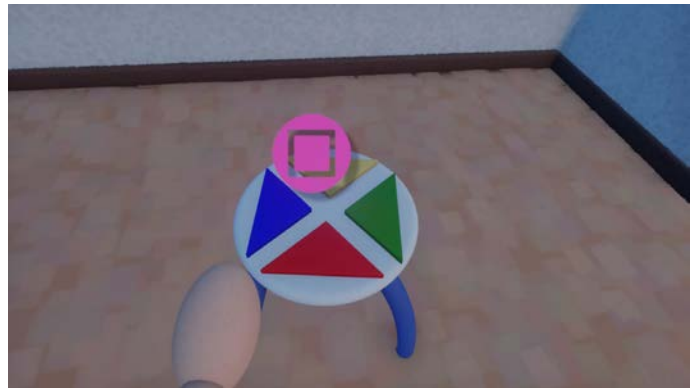
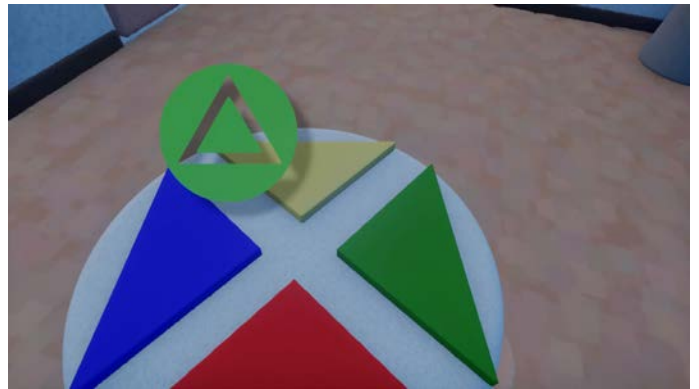
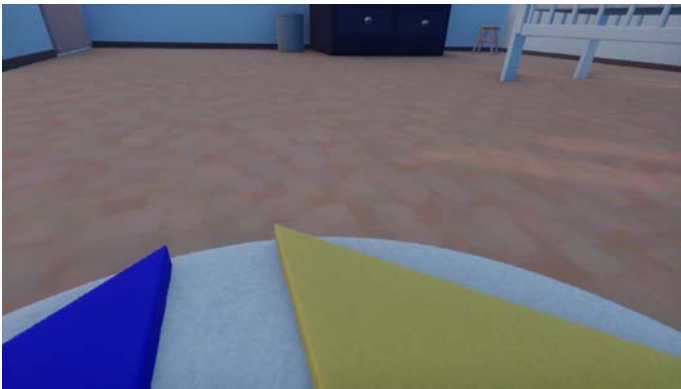
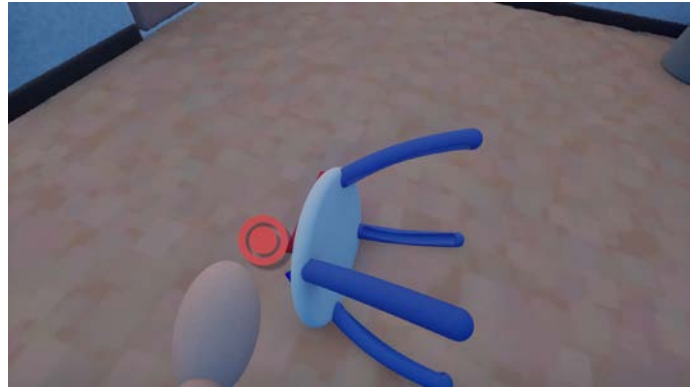
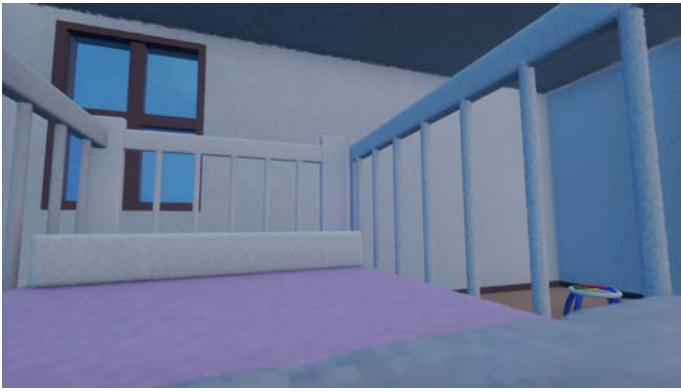


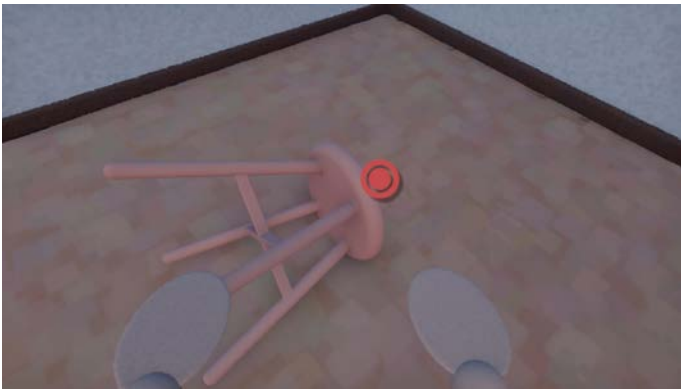
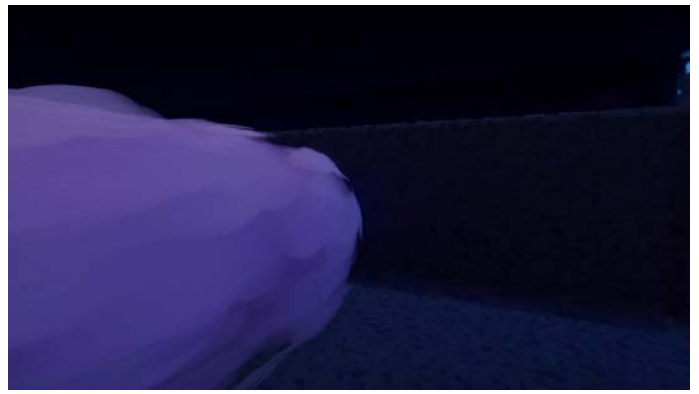
You need to clean up  
and get to bed!













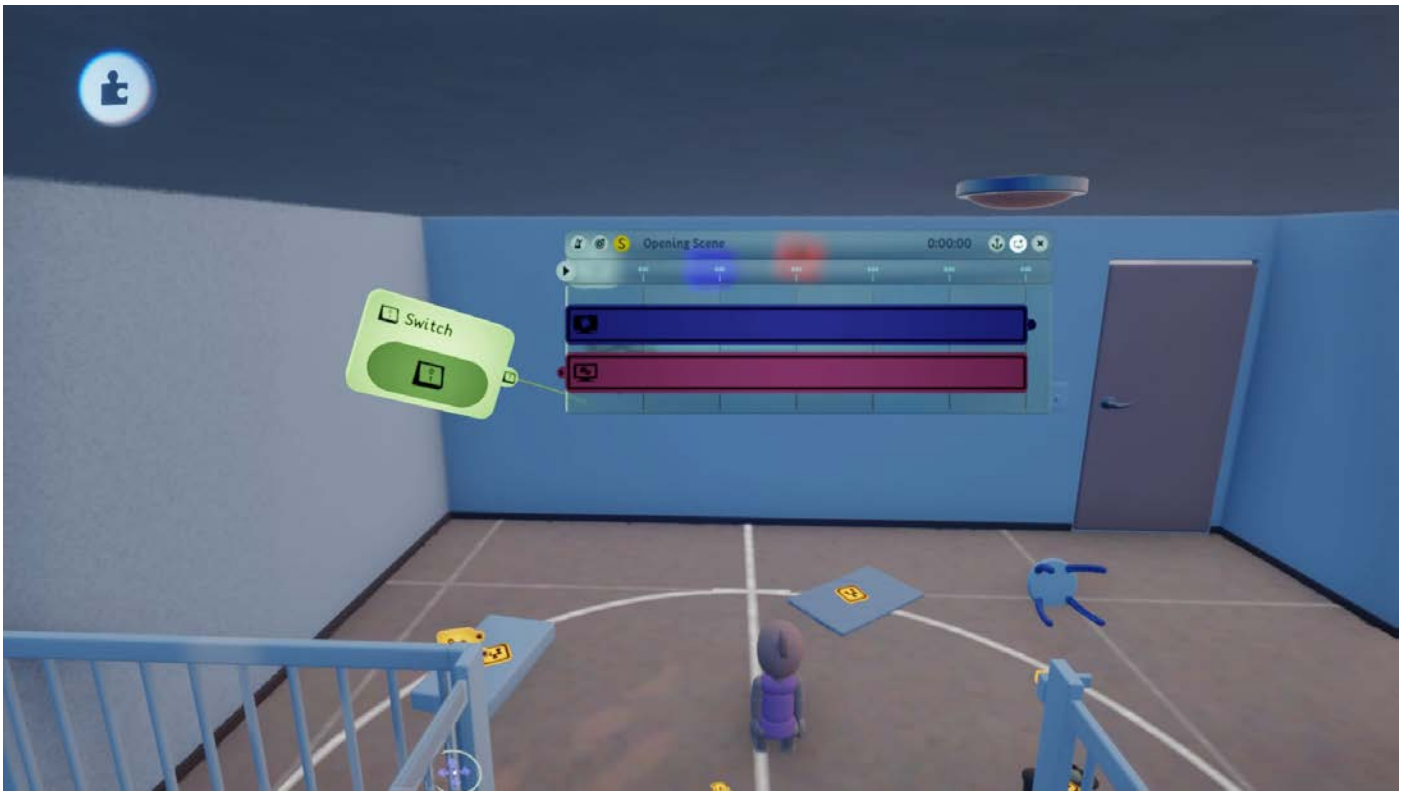


Good on you, you cleaned your room and made it to bed before pops came!



demo II

***FINAL LOGIC***



**Opening Scene**



**Pillow**



**Activity Table**



**Blanket**



Trash



Stool





**Ball**



**Drawer**



**Stuffed Animal**



**Object Locations**



**Climbing Crib**



**Crib Door**





## Latch



## Player Puppet





## Death Timer

# **CONCLUSION**

## CHALLENGES

I would like to get into some of the challenges that I faced in this project. Although it was great that the team at *Dreams* wanted to give people the opportunity to be able to make their own games on a console, it also presented some of its own problems. Using a PS4 controller, as opposed to a keyboard and mouse, was difficult.

The default control scheme made use of the motion sensor in the controller, which added another dimension to navigation. I was not used to it and did not like it in the beginning. After a while, I switched the controls to only use the thumbsticks, but it limited the navigation even further and felt more restrictive, so I switched back to the default controls.

Another issue I had was with accidentally moving elements in my scene. I would be working on one object and would be unaware that I had unintentionally moved something else. Throughout the development process and documentation, I would need to put the controller down to write various things, or pick it up to check something. In doing so there were occasions, again, where I would unwittingly move some elements. I would later notice that certain elements were off and I would have to place them back where they were supposed to go, which was an annoyance for components, which were precisely placed.

Sometimes when I would want to go back and undo or check something, I would end up undoing something else that I wanted to keep. Later, I would notice that there would be a problem with something that I fixed and I would realize that I must have undid it at some point. The positive side of that was at least I already knew what I would need to do to fix it.

A big hurdle I had to jump was changing the perspective of the player to first-person. It was one of the first tasks I did, so everything was still very new to me. I first had seen a simple way to do it through a *Dreams* tutorial, but in that method the movement felt sluggish and clunky. I went online and

searched through different people's tutorials to learn how to make a more precise FFP. In the end I was able to accomplish my goal, but it took much effort to understand each part of the logic's purpose.

A Playstation 4 is supposed to be a home console, so it is not easy to take with to travel. Creating my project on it was not very convenient because I needed my console and a television in order to work on it. It was not like working on a laptop, which is designed to be conveniently transported. This limited my options of places where I could work on my project. I knew what I signed up for when I decided to do my project on a PS4, but it was arduous at times.

The last real hurdle I had to deal with was working in *Dreams* itself. Since it was a new software, not many people had expertise in it, plus with the onset of Coronavirus, resources were limited. All of the issues I ran into I had to solve myself or by using help from online resources.

In *Dreams* you can upload your creations online and release them to the public. I planned on doing this to get feedback on parts of the game I could improve and get people's general opinions on the concept. Unfortunately, I did not receive any feedback, only a few likes, which was frustrating. I only had twenty-five people who played my first demo and two who played my second.

## FINAL THOUGHTS

Video games had been an interest of mine for most of my life. I always thought it would be a fascinating experience to make a video game. Coming into this project, I had no prior video game experience, I wanted to immerse myself and learn about the process of making a game.

I am not the best at times at game-planning and outlining the scope of my projects, usually I just want to dive right in. It helped me for this project to have some structure and proper preparation. By taking the proper steps, I was able to keep the project organized and complete the important tasks I wanted to finish.

I was new to video game creation and I was also new to *Dreams* when I started. I needed to learn how it worked on the fly. Luckily, *Dreams* offered tutorials to learn the basics, like the controls, writing logic, sculpting, physics, cameras, text, sound design, etc. Understanding the basics definitely helped me figure out and solve larger tasks I faced.

I knew that it took a lot of work to make a video game, but after trying to do it myself I have an even bigger appreciation and understanding for the people that do it. It consists of so many parts: conceptualizing, modeling, coding, sound design, testing, etc. It makes complete sense why video game studios need so much staff and resources for Triple-A games. This whole process made me realize how difficult it really is to make a video game.

The most difficult part of this project was objects or logic not acting how I intended it to act. Having to troubleshoot and adjust their settings and logic in hopes of fixing the problems became frustrating when continuous attempts would not work. At times I would be stuck not knowing what to do. Those situations reminded me of dealing with coding, like C++ or HTML/CSS.

On the positive side of the spectrum, when you are stuck on something for a while, once you finally get it to work, that feeling of accomplishment makes all the frustration go away. I definitely felt that feeling a few times, while I worked on the project. Perseverance is an important skill to have.

In the development process it would have been nice to get feedback. Both the demos I had done, I had conceptualized and been a part of the entire process. Since it was only me, there had to be plenty of issues that I did not notice or think about because I already knew the levels. I would have liked feedback with improvements, suggestions, or opinions.

The whole point of this project was to give a different perspective in relation to objects. Feedback about the players' experience with the different perspectives and how they interpreted it would have been nice. Since my game was limited to only people who have

a PS4 and *Dreams*, it made it less reachable. If I were to have made a game, which was available over more accessible platforms, I think I would have been able to get more of the feedback I desired.

In the end, I hope that the people that played my project enjoyed it. I hope that I completed my mission of giving people a different perspective of object interaction. Now that my project is uploaded online on *Dreams*, it will always be there, perhaps in the future more people will play and leave me some insights.

The demos can be played on *Dreams*, if you search for "Objective Perspective" and "Objective Perspective pt 2". They can also be found online at: <https://indreams.me/scene/dLLfwvnaATv> and <https://indreams.me/scene/dDbSwPpppYM>.

## **FUTURE ENHANCEMENTS**

I felt good about the work that I put in, but there were still aspects that I would like to improve. Although I had improved on them a bit, it would have been nice to have more detailed characters in the game. Sculpting in *Dreams* is similar to sculpting in real life, which is a skill I do not possess, so it was not easy for me to add more dimension to the characters. I would like to take the time and learn how to properly sculpt, so I can make more realistic characters. I would use this to also add finer detail to the other objects in my scenes.

I had animations in the game for the character and in the cutscenes, but I would like to improve them. I want to make the motions more fluid and natural looking, as well as more detailed. A lot could be accomplished in the cutscenes by simply adding more keyframes with action.

There are little bits of the logic that I would like to improve. I would like the player to be able to throw the object with the same button as picking it up or being able to skip cutscenes. These are minor enhancements, which I think would make a little difference for the better.



Another bit of logic I realized later that could be improved is the logic for turning the text displays on and off. I was using a selector to toggle them, but I could make it cleaner and more efficient if I would make a keyframe that turned off the text displays. I would also like to use the exclusive gates to prioritize objects when they are placed next to each other.

The microchips in my scenes are a bit messy because many of them contain a lot of gadgets. There is also a bunch of wires connecting different microchips. I want to clean up the logic, so it is more presentable and easier to view.

Another issue I had was with some objects' physical properties. Issues, for example, with the stool interacting with the player being a little finicky when being held, or the activity table needing a block to hold it up on its side, because otherwise it would fall completely over. These problems should be fixed.

Since *Dreams* would only let the users use sounds they offered on the *Dreamiverse* or sounds uploaded through a microphone connected to the PS4 controller, it was hard to get the perfect sounds that would fit the actions in my project. It would be nice to get better sound effects for the game. I want to also create the score for the levels using the software instruments *Dreams* offers.

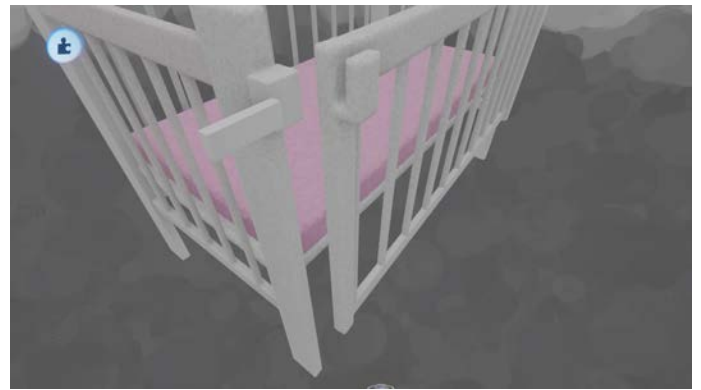
In the game I wanted the player to learn how the objects' functions through their perspective, but with objects like the latch, the player just pressed a button to open it. I would like to add a type of key command or thumbstick motion that the player would learn through the cutscene and then have to apply to the object. I feel that would better reinforce the concept of learning from the object's perspective.

The last aspect of the project that I would like to improve is the length. Right now the demos are very short, but I would like to make the first demo much longer. It would be nice to create an entire house and neighborhood. I want to have more objects for the character to interact with and more scenarios to complete. It would be great to make it into a full-game.

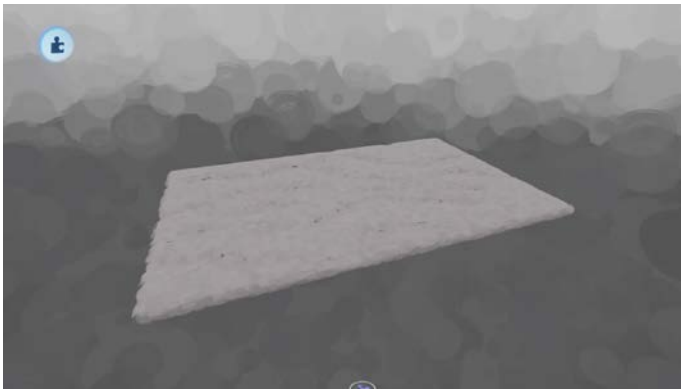
# **OBJECTS**



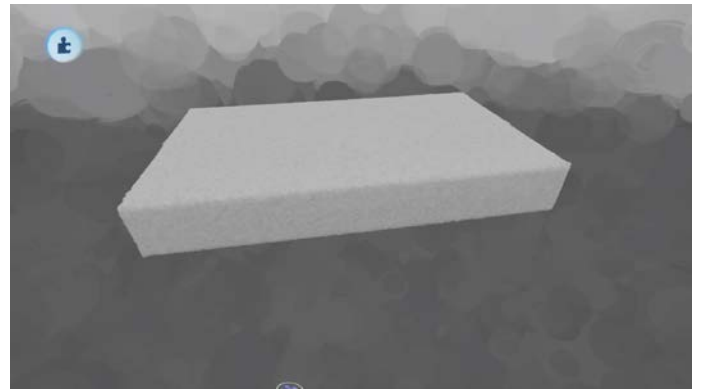
**Baby Crib**



**Crib Latch**



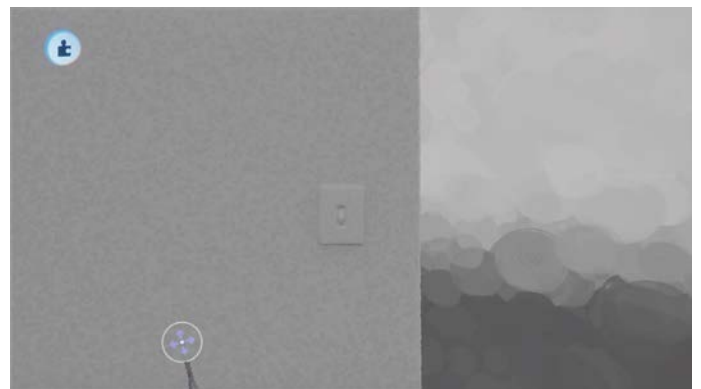
**Blanket**



**Pillow**



**Door / Handle**



**Light Switch**



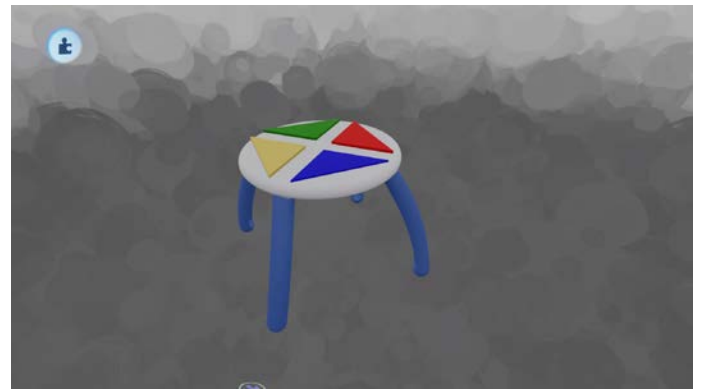
**Window**



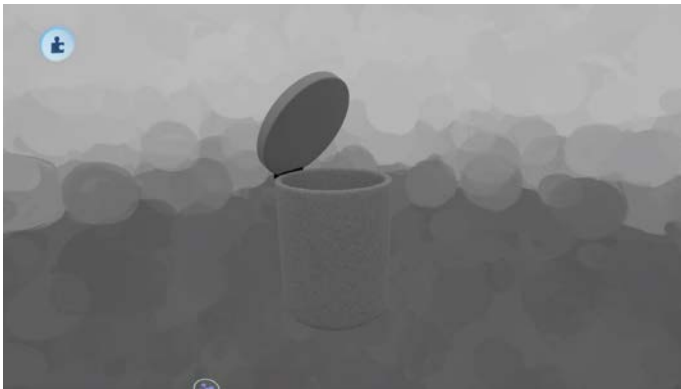
**Ceiling Light**



**Changing Station**



**Activity Table**



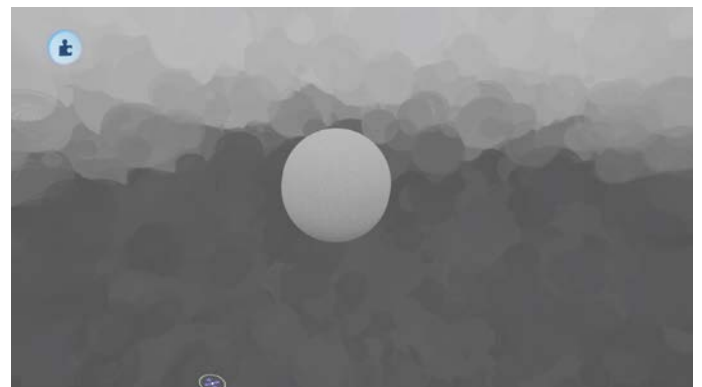
**Trashcan**



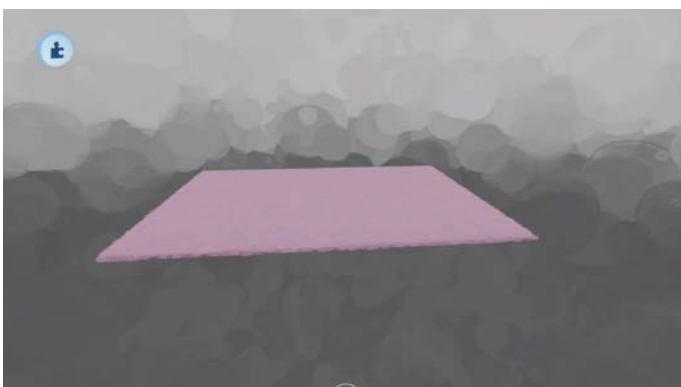
**Trash**



**Stool**



**Ball**

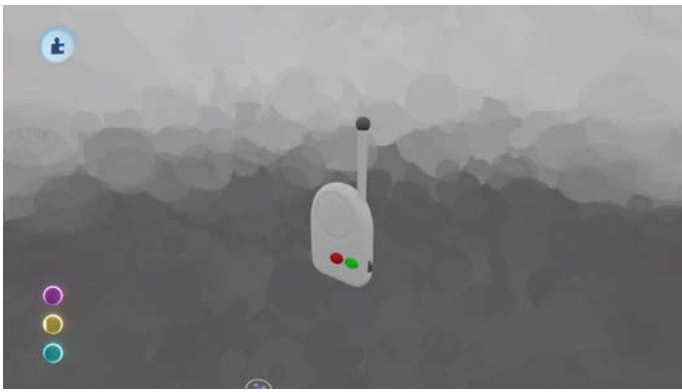


**Rug**

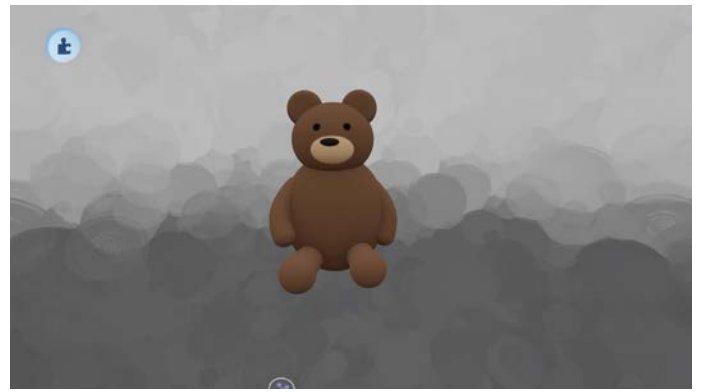


**Desk Lamp**





**Baby Monitor**



**Stuffed Animal**



**Dad**



**Baby**

# ***WORKS CITED***

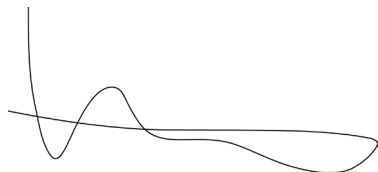
- Adolf, K. E. "Motor and Physical Development: Locomotion." *Encyclopedia of Infant and Early Childhood Development*, Edited by Marshall M. Haith and Janette B. Benson, Academic Press, 2008, pp. 363–371.
- Bernstein, Douglas and Peggy Nash. "Chapter 9: Human Development." *Essentials of Psychology*, 4th ed., Houghton Mifflin, 2008, p. 350.
- Entertainment Software Association. "2020 Essential Facts About the Video Game Industry." 20 July 2020, p. 4. [https://www.theesa.com/wp-content/uploads/2020/07/2020-ESA\\_Essential\\_facts\\_070820\\_Final\\_lowres.pdf](https://www.theesa.com/wp-content/uploads/2020/07/2020-ESA_Essential_facts_070820_Final_lowres.pdf).
- Gibson, James J. "Chapter 8: The Theory of Affordances." *The Ecological Approach to Visual Perception*, Psychology Press, 2015, p. 119.
- Grubb, Jeff. "March 2020 NPD: Animal Crossing Powers March to Blockbuster Game Sales." *VentureBeat*, VentureBeat, 24 Apr. 2020, [venturebeat.com/2020/04/21/march-2020-npd-animal-crossing-powers-march-to-blockbuster-game-sales/](https://venturebeat.com/2020/04/21/march-2020-npd-animal-crossing-powers-march-to-blockbuster-game-sales/).
- Hanson, Ben. "Why Media Molecule Wanted To Make Dreams." Interview with Mark Healy, and Siobhan Reddy, *You Tube*, Game Informer, 23 Oct. 2018, [www.youtube.com/watch?v=rq3cacfA-yE](https://www.youtube.com/watch?v=rq3cacfA-yE).
- Murkoff, Heidi and Sharon Mazel "Chapter 6: Behavior." *What to Expect the Second Year: From 12 to 24 Months*, Workman Publishing Company, 2011, p. 176.
- Norman, Don. "Chapter 1: The Psychopathology of Everyday Things." *The Design of Everyday Things*, Revised and Expanded Edition, Basic Books, 2013, p. 11.
- Piaget, Jean. "Chapter 19: The Origins of Intelligence in Children." *The Essential Piaget*, Edited by Howard E. Gruber, and J. Jacques Vonèche, Basic Books, 1977, pp. 216–218.
- Shanley, Patrick. "Nielsen Reports 45 Percent Spike in U.S. Video Game Usage." *The Hollywood Reporter*, 7 Apr. 2020, [www.hollywoodreporter.com/news/us-video-game-usage-up-45-percent-nielsen-reports-1288738](https://www.hollywoodreporter.com/news/us-video-game-usage-up-45-percent-nielsen-reports-1288738).
- Vogeley, K., et al. "Neural Correlates of First-Person Perspective as One Constituent of Human Self-Consciousness." *Journal of Cognitive Neuroscience*, vol. 16, no. 5, 2004, p. 819., doi:10.1162/089892904970799.
- Zeveloff, Samuel I. "Chapter 4: Form and Function." *Raccoons: A Natural History*, Edited by Bolen, E. Anne, Smithsonian Institution Press, 2002, p. 70.

## **Declaration of honour**

I hereby declare on my honour that I have prepared this work without improper help from third parties and without using any other means than those indicated. The information, methods and concepts taken directly or indirectly from other sources are identified by reference to the sources. This work has not been submitted to any other examination authority, either in Germany or abroad, in the same or similar form.

I affirm in good faith that, to the best of my knowledge.

I have told the whole truth and have not concealed anything.

A handwritten signature in black ink, consisting of a vertical line on the left, a small loop, and a long horizontal stroke that ends in a small loop on the right.

Any other information relevant to the examination is to be added, if applicable.