Claw Machine

A blender game
by Chan Hiu Yan (55131678)


Score: 0
Coin: 3
15
SPACE
C
Watch demo video: https://youtu.be/vJt868HBm20

Contents

Hello, I'm Grace!

- Year 3 student in BAS Creative Media, City University of Hong Kong
- First-time Blender user
- Experience with Processing, p5.js, a little Maya

About the Game

The Claw Machine game is made purely with Blender, both modelling and logic. It uses Blender Game Engine and heavily relies on the physics simulation provided.

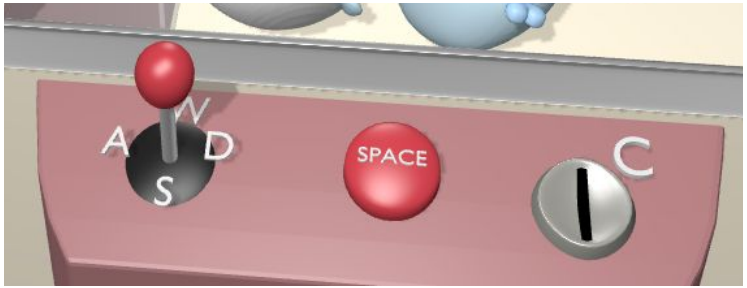
Why I chose the idea of a claw machine?

I wanted to make something simple - a nice game. The claw machine game is a familiar game to many, easy to play, and yet has its own complexity. Each round has countless possible outcomes, presenting a challenge to master.

How to Play

The game is made to be self-explanatory without an instruction page.

1. C key to insert a coin
2. Maneuver the claw with W - up, S - down, A - left, D - right.
3. SPACE key to watch the claw go down

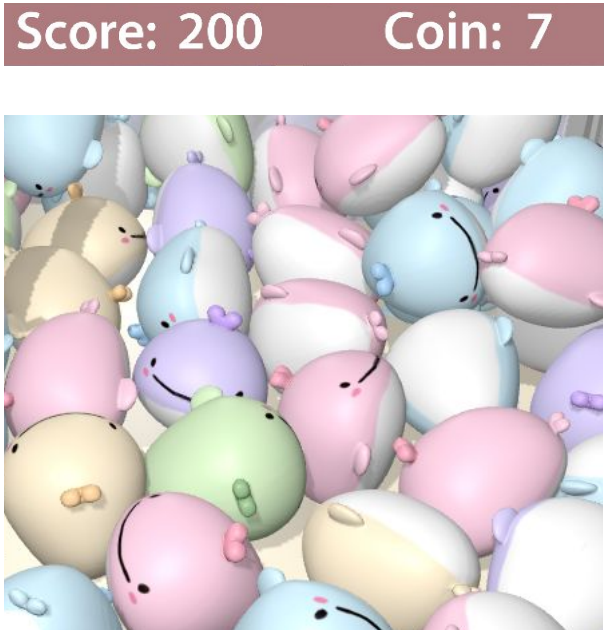


How to Play

Each plush is worth a different number of points.

- Purple: 1000
- Green: 500
- Yellow: 200
- Pink: 150
- Blue: 100

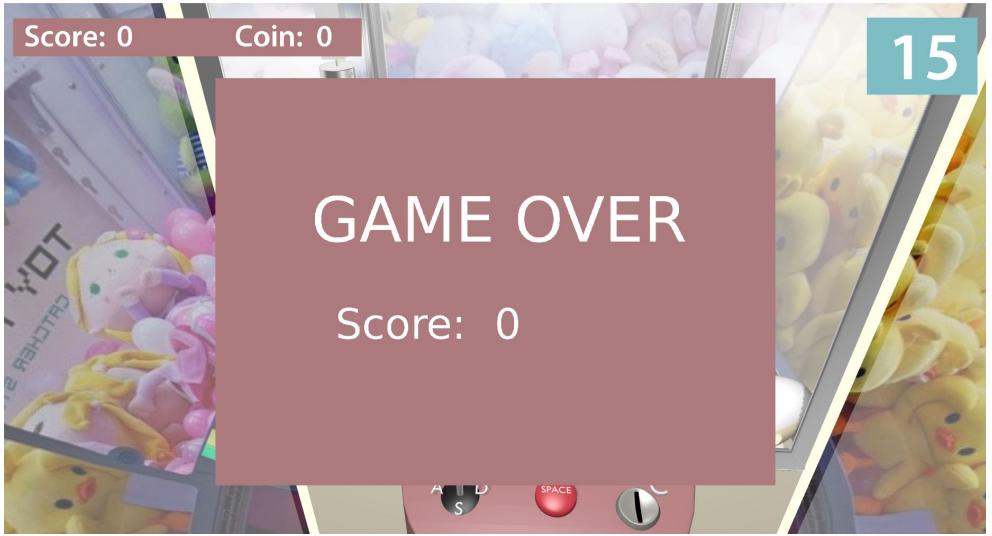
Getting a plush will also increase your coins.



How to Play

Each insert of coin gives you a countdown of 15 seconds to maneuver to claw. When the time is up, the claw automatically lowers down.

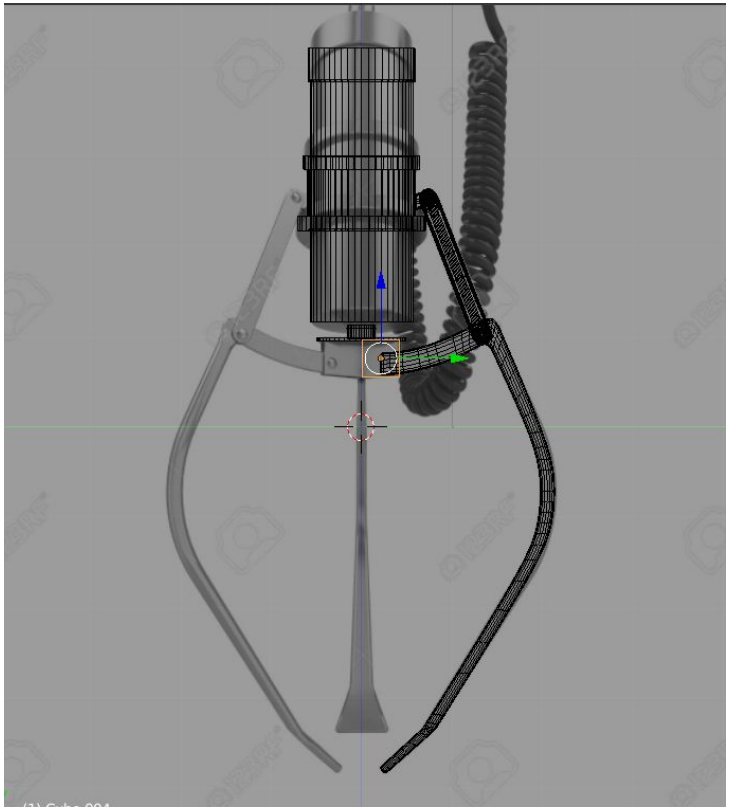
The game is over when you run out of coins. The goal of the game is to gain a high score.



Making Process

Modelling the claw

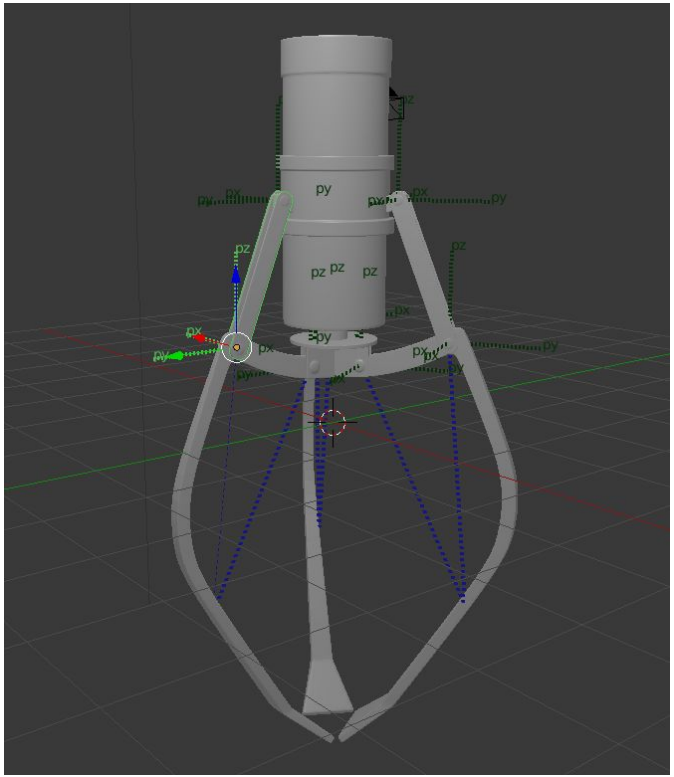
I separated the different parts of the claw as different objects, so they can interact in physics.



Making Process

Modelling the claw

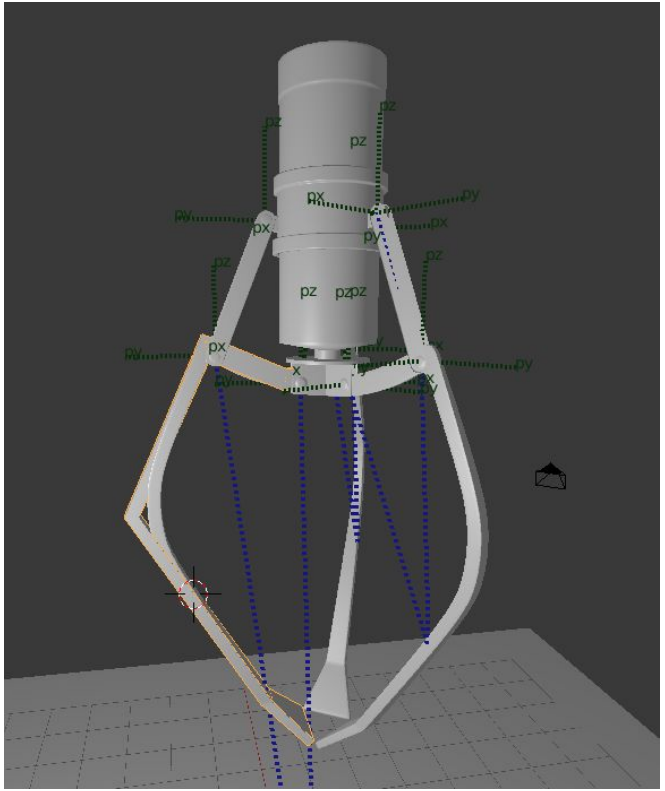
I used rigid body joint hinges to relate the claw parts. The sliding of the middle cylinder will control the opening and closing.



Making Process

Modelling the claw

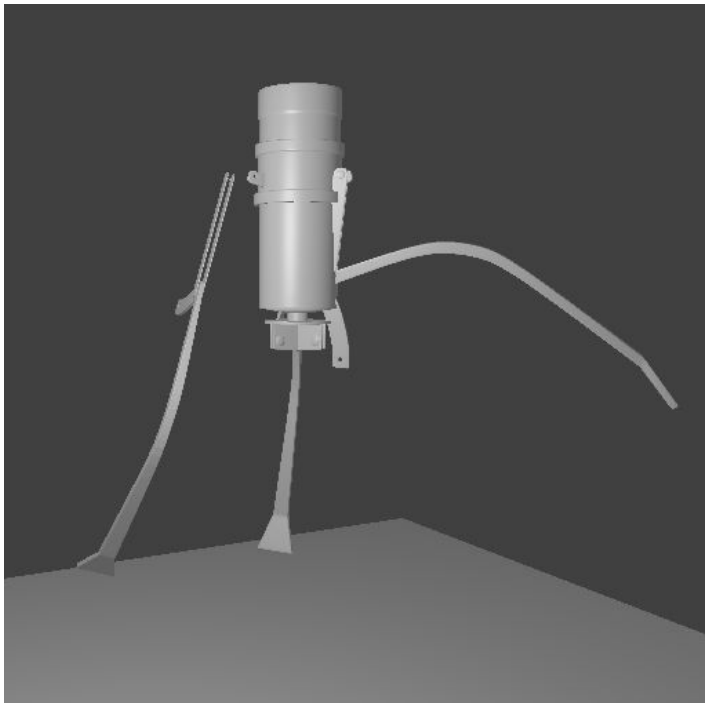
When testing with picking up spheres, the claw goes through the objects because there is no collision. I added a low-poly hook for collision detection that is parent to the high-poly claw for display. I also added a low-poly cylinder to control the main cylinder.



Making Process

Modelling the claw

Meanwhile there were a lot of overreactions from the claw collisions and restraints. These were painstakingly fine-tuned with trial and error and googling. I added a maximum angular velocity to sooth the symptom.

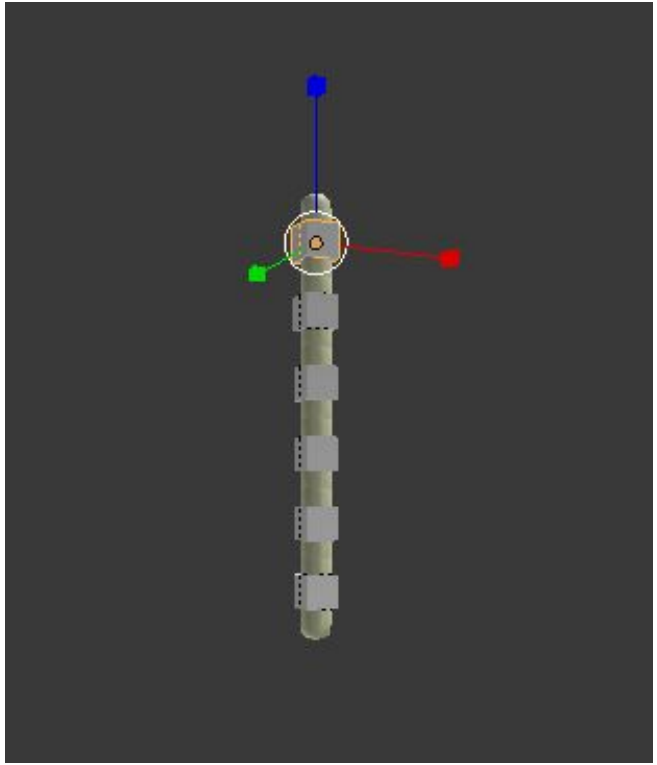


Making Process

Modelling the chain

I wanted the claw to have a swaying space so that 1) it is more realistic, 2) it is not forced to lower into the ground/objects but has room sway aside.

I modelled it with a series of rigid body joints.

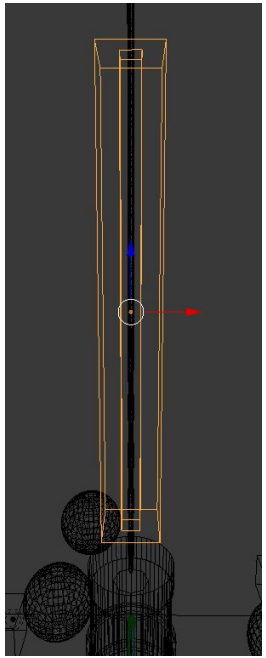


Making Process

Modelling the chain

The problem with the chain is that it takes too long for the top movement to ripple down to the claw. It swings like a slow pendulum.

I added a tube to constrain it.



Making Process

Adding the logic

To prevent the claw from moving out of the box, lowering through the ground, or up the sky, I must constrain the position.

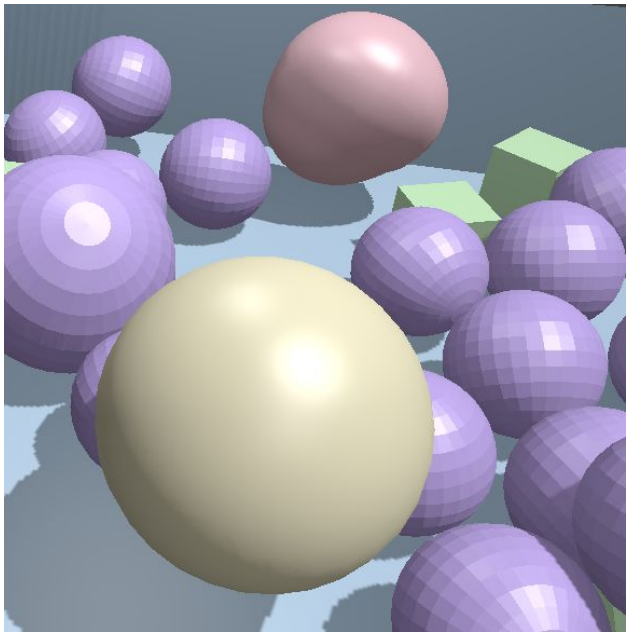
I couldn't find a property that does that, so I googled and used this python code.

```
import bge

def getZPos():

    cont = bge.logic.getCurrentController()
    nozzle = cont.owner

    nozzle["z_pos"] = nozzle.localPosition.z

    print(nozzle["z_pos"])


getZPos()
```

Making Process

Modelling the plush

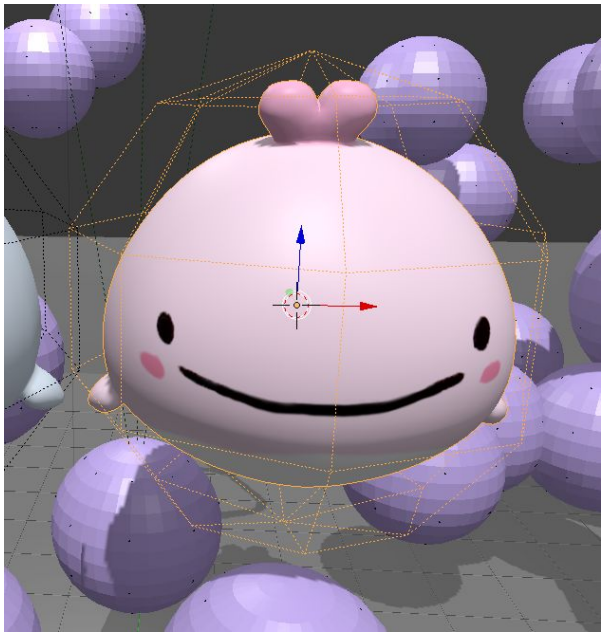
I tried so many ways to make a soft pillow-like plush, but it just didn't work. Soft bodies looked like fluid blobs and cannot be picked up. I tried parenting the soft body to a rigid body, which looked like an energetic water balloon with a tumor. After all, I resorted to the rigid body.



Making Process

Modelling the plush

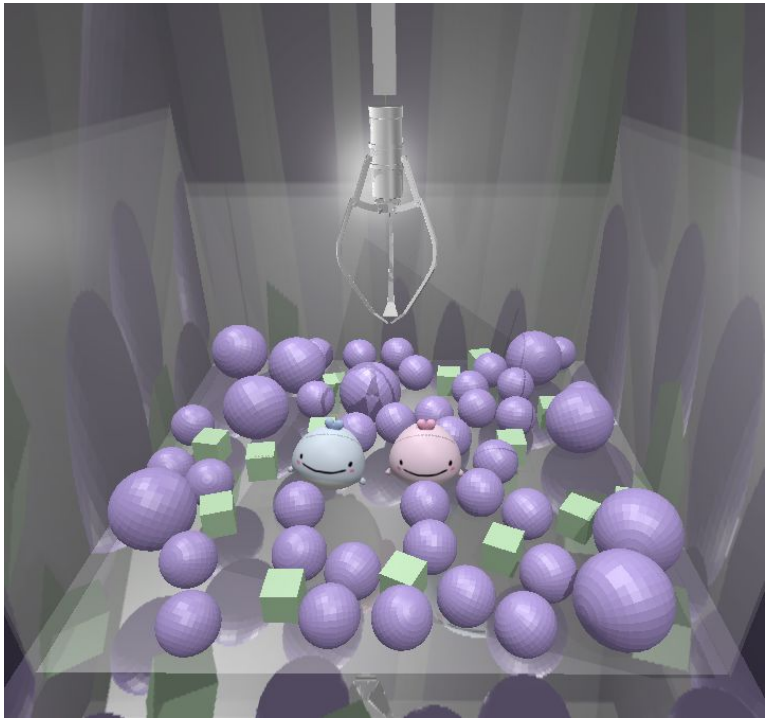
I modelled a simple whale plush from a cube. The UV map was exported and painted in Photoshop. I used a sphere-shape collision bounds for ease of calculation.



Making Process

Modelling the glass

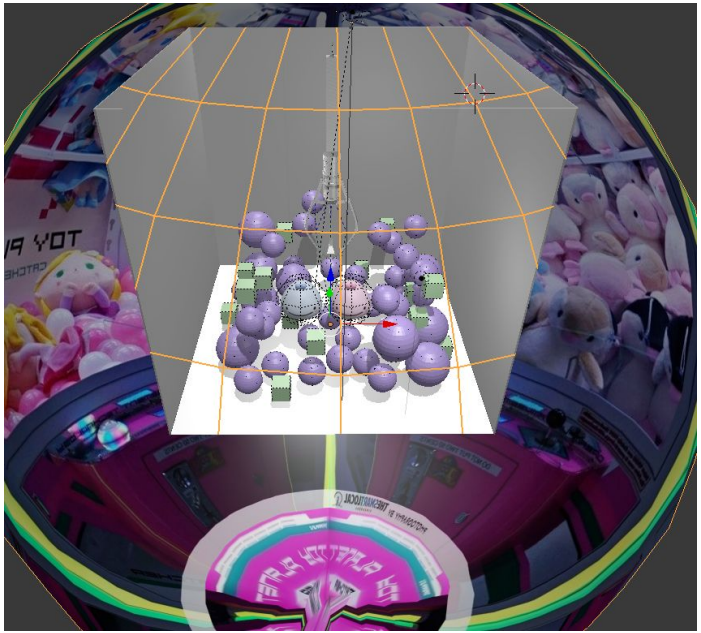
I wanted the glass windows to reflect, so I looked up and found a code for that. The effect was not as desired, so I stuck with semi-transparency for the glass.



Making Process

Modelling the environment

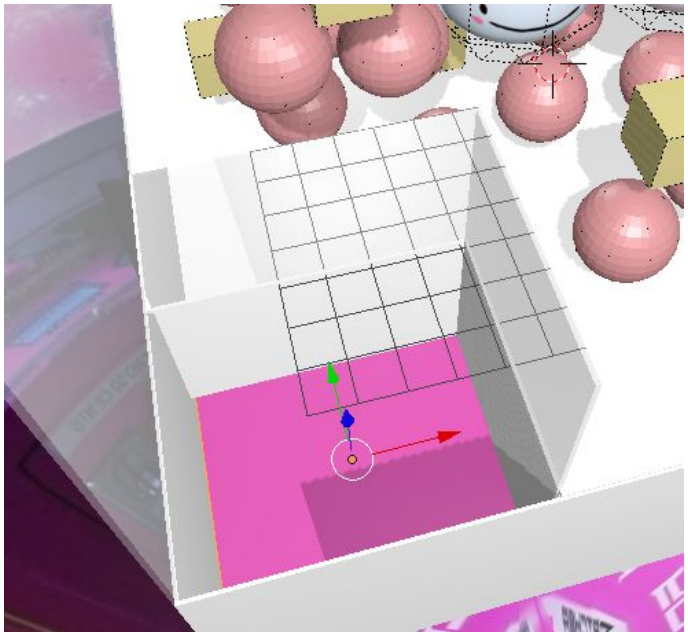
For the environment to look like clawing machines without making two additional ones on each side, I used a large sphere with UV mapping.



Making Process

Adding the plush sensor

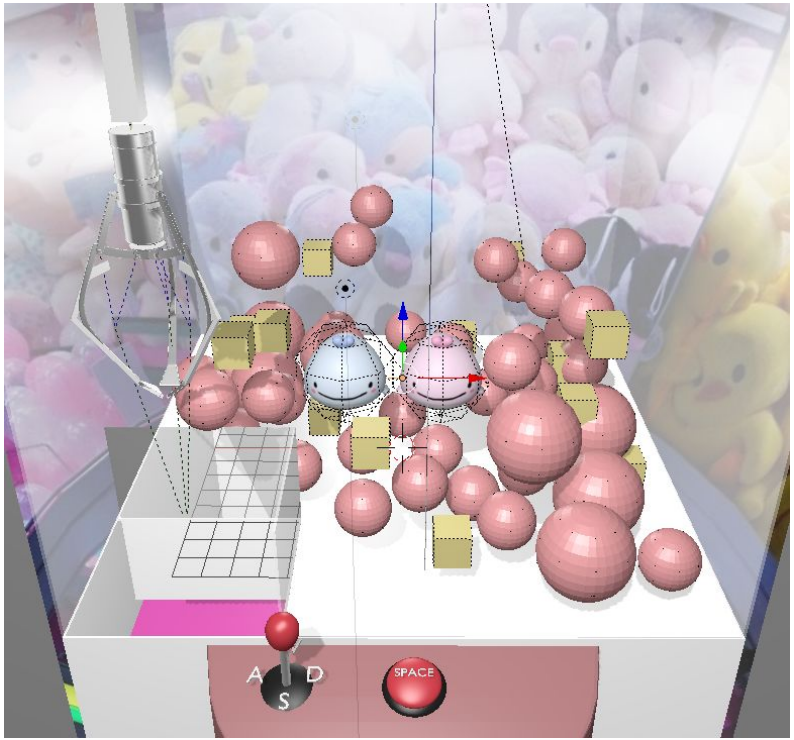
Initially I used a ray sensor, but I noticed some plush pass through undetected. Later, I switched to a radar sensor. With python, the sensor will know which object is hit.



Making Process

Manipulating the camera

For the camera to strictly follow the claw makes the view enter the box, and also moves too much left and right. I ended up making the camera follow an empty, whose z-position is fixed on the floor, with its x, y-positions being 1/3 that of the claw.



Making Process

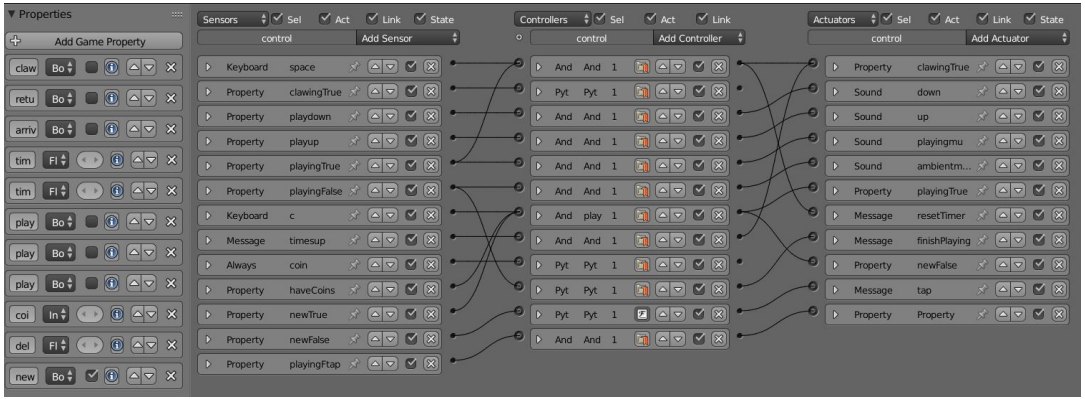
Animating the controls

Each control plays an animation when activated. The main logic is linked elsewhere, to the pink box beneath it.

Making Process

Coding the logic

Most of the logic is linked to the pink box as python and brick codes. This controls whether the player has inserted a coin, whether time is up, etc. I will not go into details here.



Making Process

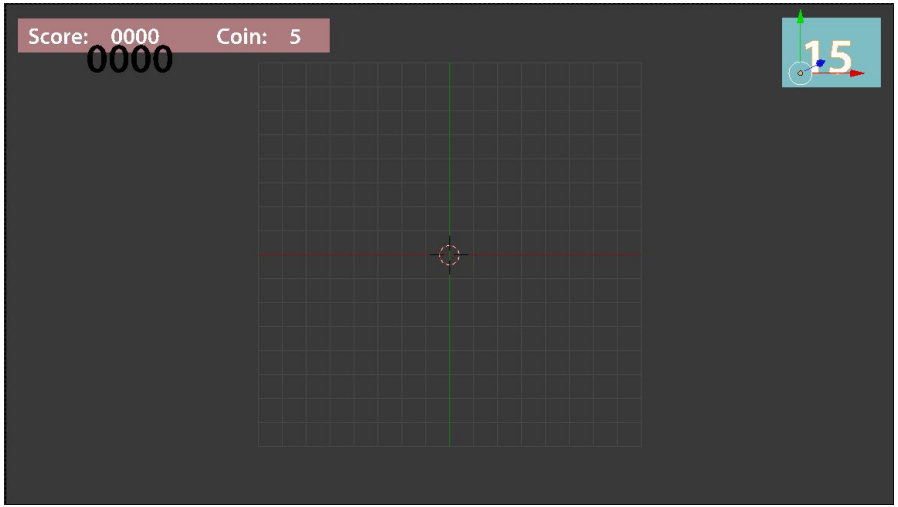
Adding sound

At first I found claw machine music that were from western arcades. When I searched in Chinese, I found something that I hope would be more familiar to Hong Kongers. I also trimmed other sounds from claw movements and coin.

Making Process

Adding the overlay

I used another scene as an overlay. The scores, number of coins, timer are displayed here. Each has its logic bricks, python, properties, and messages to interact with the pink box.



Making Process

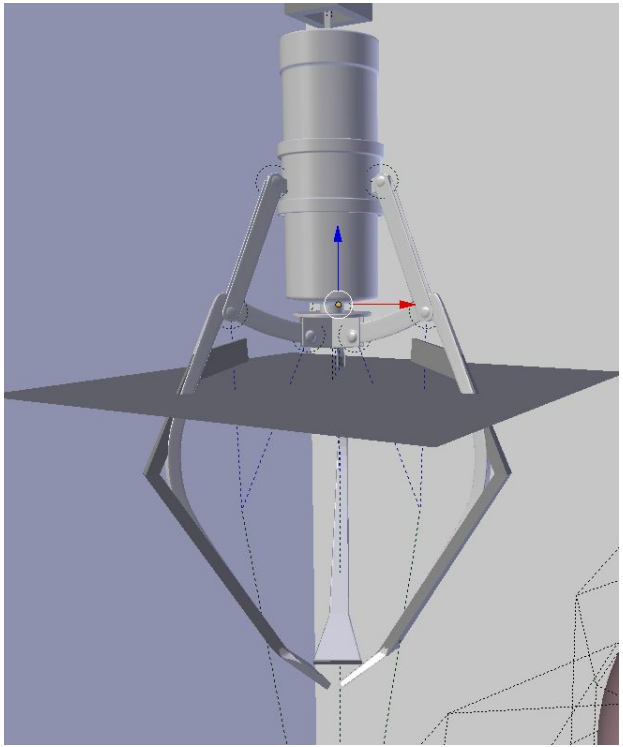
Adding the gameover

When the player is 1) out of coins, 2) some delay after finished clawing (perhaps the player earned coins with the last coin), the game is over. When the game is over, no more coin insert is allowed.

Making Process

Adding detectors

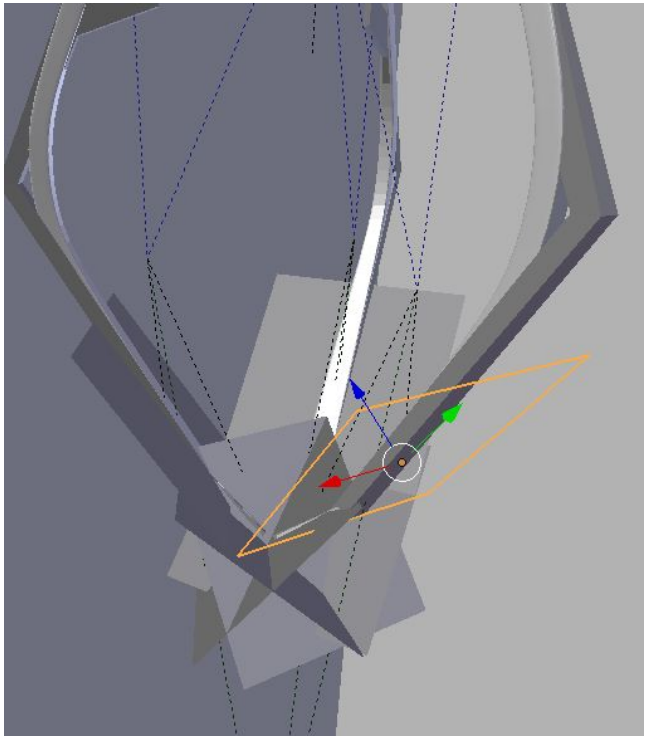
The claw will stop lowering when it reaches the floor. However, if it hits a top layer of plush it should also stop before it overreacts. I added a ray sensor to detect approaching plush.



Making Process

Adding detectors

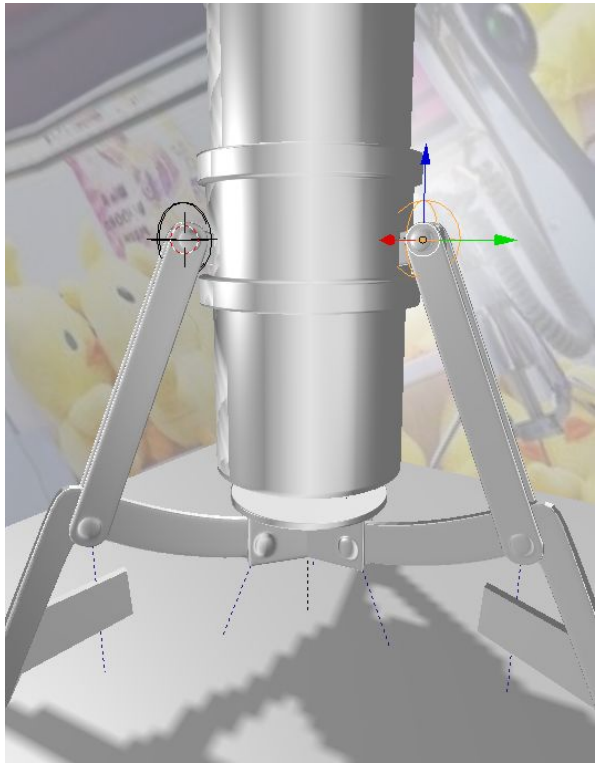
One problem I constantly face is the overreaction of the claw when it forcibly open or closes upon collision. Some collision is desirable (holding a plush in grasp) but not when it rips the claw apart. I ended up adding 3 ray sensors to each leg. When it is near another object, it refrains from opening or closing.



Making Process

Reset mechanism

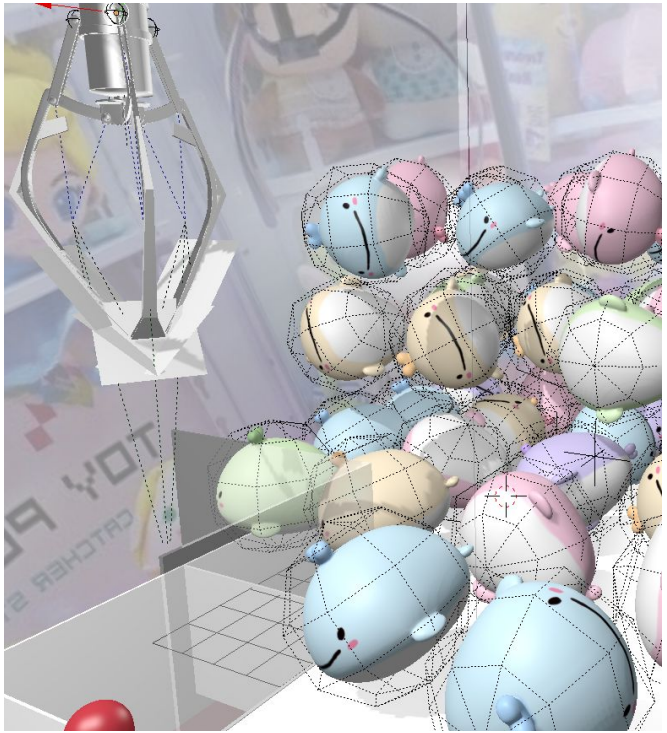
Better, but still easily overreacting. I decided to make the claw positions and orientations reset if they are far from where they should be. I added empties parented to the claws. If these empties are far from the objects that holds them, reset. This was done with python.



Making Process

Initializing the plush

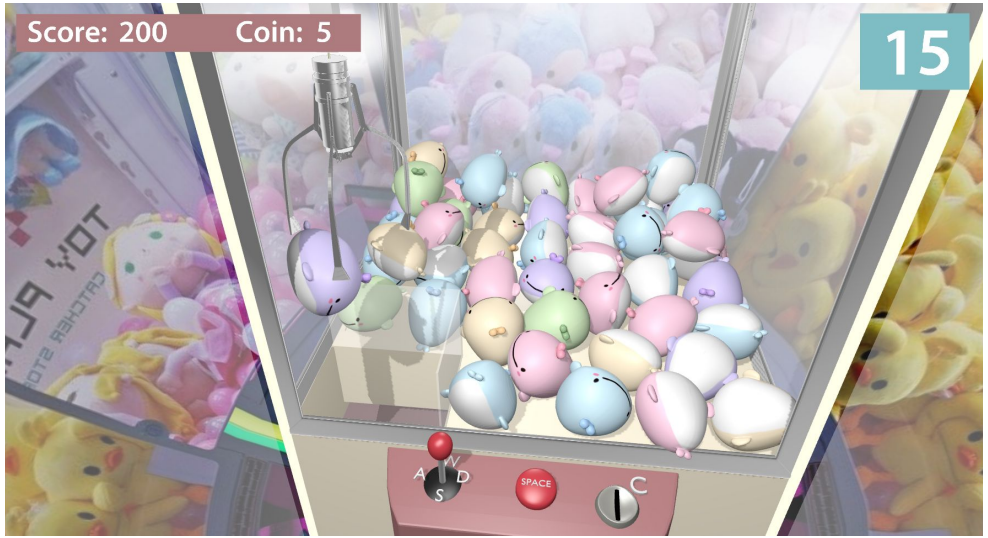
I duplicated the plushies across the box in three layers. I set the worth properties of each color differently so the points earned would differ.



Making Process

Adding final touches

To finish off, I added the metal frames and light-up bars on the edges. I also cleared up unused files and debugging functions.



Making Process

Exporting

I followed given export instructions, which originally gave an application that ran all pink. I found out I must pack the external data before export.

And ta-da! I am pleased.

Further development

Some areas for possible development:

- Startup menu, high score, play again button
- Randomized plush positions each round
- More variety of plush
- Ragdoll plush
- Further fix on the overreactions
- Bonuses, special superpower plush
- Hall of plush collection
- Level design