Gluttony Run

By Ryan Smith, Tim Phillips, Shane Kusmierz

# Table of Content

Scope Pg3

Character State Pg4

Uml Diagram Pg 5

Team Progress Pg6

Shane Kusmierz Pg6-8

Ryan Smith Pg9-Pg10

Timothy Phillips Pg11

# Scope

* The player will be able to switch between three different states a fat state that can barely jump and is slow but can charge through walls, a fit muscular state that is safe from both the extreme ends and gives the most points but gets past the least amount of obstacles, and a lean state that jumps the highest and runs the fastest but is in constant danger from starving to death.
* When the player gets too fat or too lean they will die forcing the player to balance between healthy and unhealthy foods, on top of that there will be a point bonus for eating foods matching the state you are in tempting the player with a risk vs reward idea.
* Both the fat form and the lean form will have an obstacle they can get over without leaving their lane, the fat character will be able to bowl over the steel walls and the lean character will be able to jump far over the piles of rubbish in the road.
* When the player eats enough food they will be able to sail through the sky with a super fart attack allowing them to go past a bunch of obstacles easily.
* Have a fat bar that shifts back and forth depending on how much bad/good food the player has eaten, or if the player hasn’t been eating any food at all.

# Mechanics

## Character States

The character is programmed to move differently in the three states they are in. the code for the muscular state is:

“ if (PlayerRun.fat > 0 && PlayerRun.fat < 600)

{

fatRunning.SetActive(false);

skinnyRunning.SetActive(true);

modelNumber = 2;

playerScript.speed = 4;

//print("Model 2 : Muscle");

}”

What this is saying is that when the player is between 0 and 600 on the fat bar the code will set them to the model number 2 which is the muscular model, it will accelerate at 4 units and use the skinny running settings to make it look like the guy is in shape.

When the player is lean the code is:

else if (PlayerRun.fat > 0)

{

skinnyRunning.SetActive(false);

strongRunning.SetActive(true);

modelNumber = 3;

playerScript.speed = 5;

// print("Model 3 : Lean");

}

This sets the player to the fasted movement which is an acceleration of five units, and it will use the third model which is the skinniest looking model. The code will activate only when the fatbar dips below 0 units.

The final state is for the fat state:

else

{

strongRunning.SetActive(false);

fatRunning.SetActive(true);

modelNumber = 1;

playerScript.speed = 2;

// print("Model 1 : Fat");

}

Ryan set this one to be an else statement because it was the only option left for the code after the other two states are cycled through. It’s the slowest state only accelerating 2 units at a time and uses the default model 1 since that is the one the player starts with.

## Movement

The character is programmed to move differently in the three states they are in. To get the states to move we have the formula “pos.z += speed \* Time.deltaTime” in place. The value of speed is different based upon which of the states the player is in. The fat state will have the lowest value the muscular state the second biggest, and theleanest will be the fastest.

## Jumping

As with movement jumping is going to be different with each of the states. To make them jump the formula is “rb.velocity += Vector3.up \* jumpForce”. The value of jumping will be different based upon which of the states the player is in.

. The fat state will have the lowest value the muscular state the second biggest, and the leanest will be the fastest.

## Fart

The fart is the players special ability and will only be active. The players fart bar fills up naturally over time very slowly with this:

if (fart < 1)

{

fart += .001f;

fartReady = false;

}

else

{

fartReady = true;

}

The fart meter slowly ticks up over time and when it gets filled up finally they will be able to activate the ability and fly over all obstacles on the map.

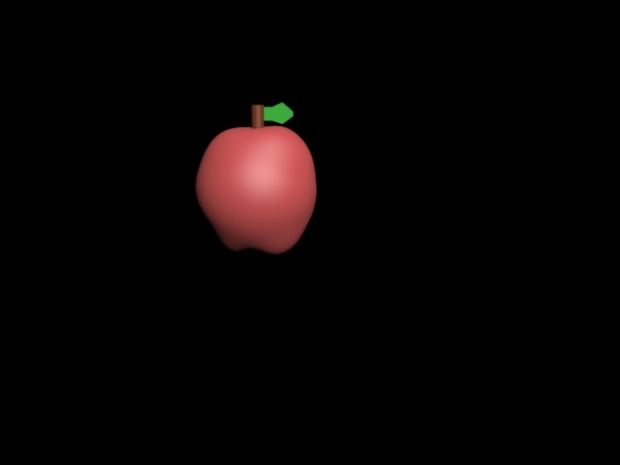
# Game and Teams status

## Game project

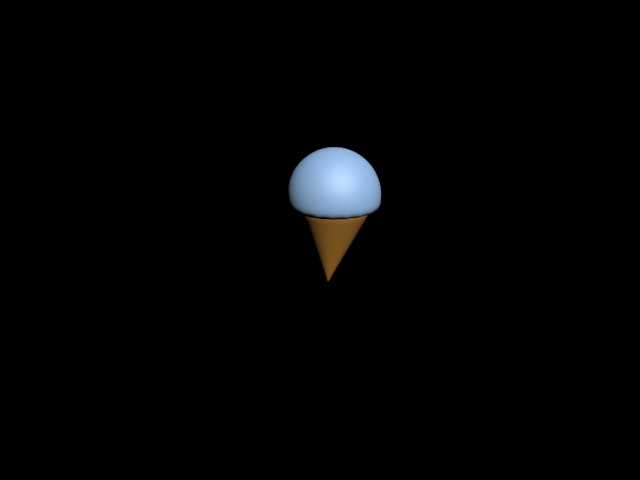
The games current status is at around 60%. We have everything able to spawn currently, the three food items and the three obstacles. The player spawns and is able to switch in between the three different objects. The UI elements work and are able to fill the indicator for farting as well as shift the indicator of the fat meter.

## Shane Kusmierz

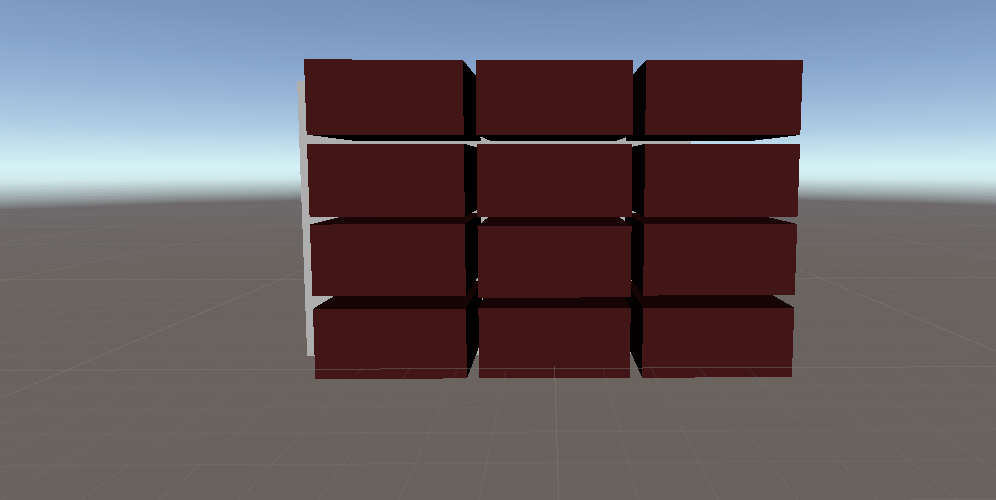
For My job I oversaw the environmental art like the food and background designs. I am also in charge of the voice acting for the announcer and how the food interacts with the player. I have modeled the food, muscle powder, and ice cream models along with the obstacles for steel walls, brick walls and garbage can. I have also coded the interactions with game elements such as implanting the script that destroys food objects. I also am in charge of the design documents formatting, length, and stylistic choice. I have also placed a script in all of the lanes to make said obstacles spawn.



The apple model



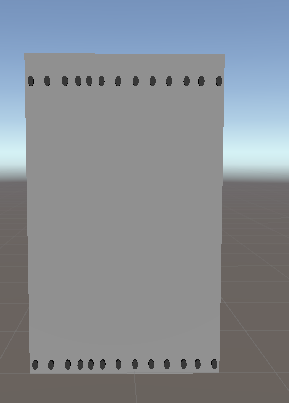
The ice cream model



Brick wall model



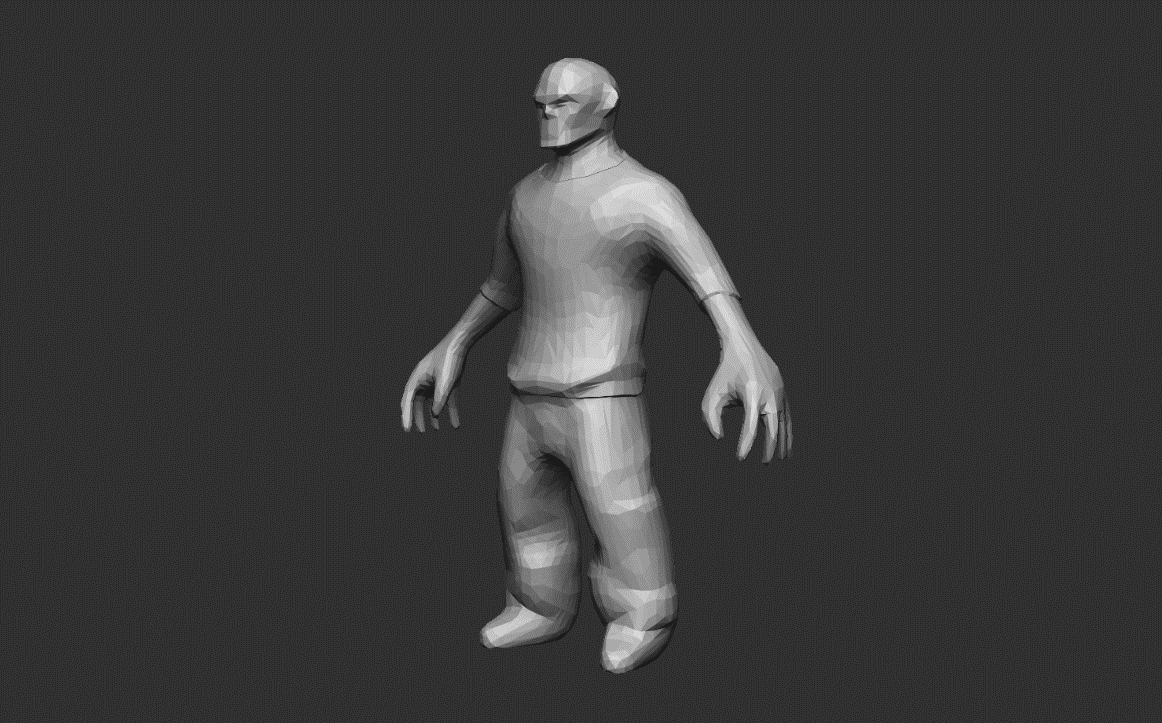
Muscle powder



Steel wall

## Ryan Smith

Ryan Smith is the guy in charge of the characters. This means that he is in charge of the mechanics, animation, and rigging along with the voice acting for the character. Some of the things he has made progress on was modeling the characters three different state, along with making sure the character does not fall through the floor. He alongside Tim also made sure that the UI elements for the character corresponded with which state the character in so when the Fat bar is leaning to the right the character is skinny and as it progresses more towards the left going to muscular and then eventually fat. He also has worked with the food destroyed code I made and has it so that now when the food is passed by it dies after a period of time.



Skinny guy model



Strong guy



Fat guy

## Timothy Phillips

Tim is in charge of all the UI elements and the background music along with being the project manager. What this means is that Tim has programmed the score making it so that it increases slowly as the player progresses through the level, the fat bar making it so that the player’s fat bar decreases overtime and the state changes to reflect that. The fart



Create super class for all gameObjects except a few

All gameObjects inherit the super class

In the super class’ update:

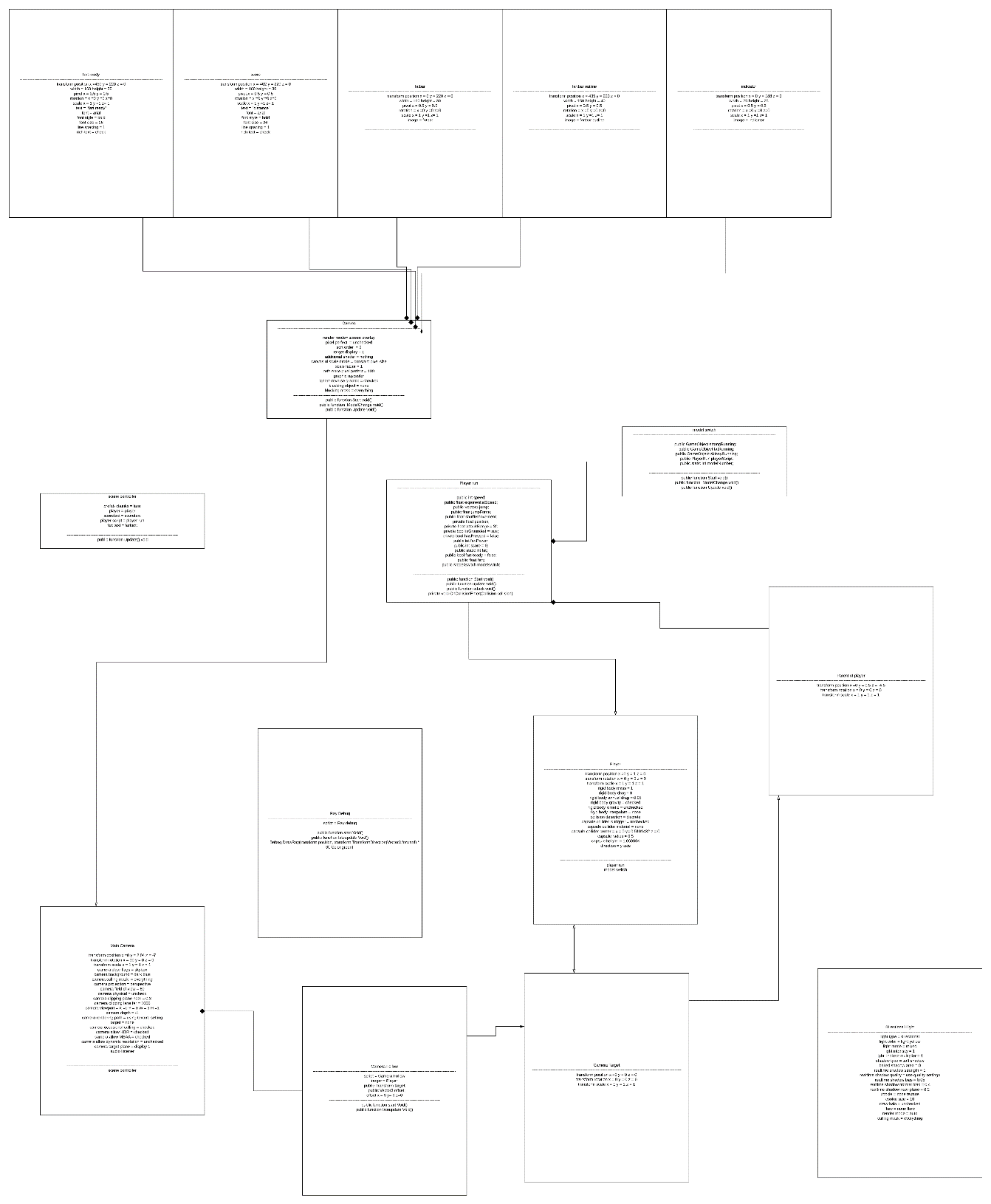
Retrieve data from player components as necessary

Update this object depending on player component.

# State Machine

# 

# Uml Diagram



# Discord link

<https://discord.gg/ssgEaXc>