CHICKEN_GAME

STUDENT: GREGOIRE COUSIN, SUDENT ID: 18204188
STUDENT EMAIL: GREGOIRE.COUSIN@UCDCONNECT.IE
COMP30540 GAME DEVELOPMENT ASSIGNMENT

INTRODUCTION

In our GAME DEVELOPMENT class, we were assigned the task of building a two dimensional video game with the purpose of learning to interact with and understand the features of Game Maker 2. The project consisted of manipulating a variety of images and their associated objects within a room. I was able to create and move constraints in my code to make sure certain objects weren't inhibiting the intended behavior of others. I acquired the ability to morph images, to change their dimensions and to create good flow when it came to switching images based on others they were colliding with. I also learned how to interact with the Game Maker UI and discovered a number of tools that would provide for expansion of the game submitted.

OBJECTS

1.obj_drop object.

This object is the core mechanism/function for the chicken eggs to drop in a random manner from a desired space within my room. The intention of building an object without any type of visible sprite placement within my room at game start was to learn how to randomize eggs dynamically from the top-down and subsequently to worry about manipulating their position based on other apparent objects. This was a good and necessary challenge, allowing me to expand my game in the way that I wanted and to also relieve CPU and memory load based on the approach. As you can see from figure 1, I had a "global var"; it's function was to increase the number of eggs from the top of the room down. The object also relates the position of the egg to the nearest resting chicken at the start of the game so that when the egg is prompted to come down, the chicken will 'wake up' for a determined time period and will then go back to its root/default state. I also ensured that this object was capable of dropping a desired amount of eggs faster than hay. To do this, I created a random variable ranging from 1-10 asking the object to trigger three quarters/75% more eggs at game start. This also gave me the opportunity to create a variable to change the amount of hay released throughout the game so it would become more challenging for the player. It can also be noted that an alarm is triggered at 1 second of the frame's steps to drop the items in accurate time. Figure 2 of obj_drop's relation is based on how the dropping of items will react to the game over time. "counterEGG" enumerates the eggs that are dropped and is used not only to display the number to the player but also to change the behavior over time. After the basket catches

```
if num<gamespeed{
  if r>=1 and r<=8{
    xp = irandom_range(40,1000)
    yp = 100;
    makedrop_e=obj_egg
    var chicken = instance_nearest(xp,yp, obj_chicken)
    dropegg = instance_create_layer(xp ,yp,"instances",mak
    chicken.sprite_index = spr_chick_pound
    alarm[0] = 1*room_speed; // call alarm 0 every second
    dropegg.speed=8;
    dropegg.direction=270;
    dropegg.image_xscale =0.8;
    dropegg.image_yscale=0.8;
    dropegg.gravity = 0.2
  } else if r>=hayspeed {
    xp =  irandom_range(40,1000)
    yp = 100;
    makedrop_h=obj_hay
    drophay = instance_create_layer(xp,yp,"instances",make
    drophay.speed=8;
    drophay.direction=270;
  }
}
```

figure1: obj_drop

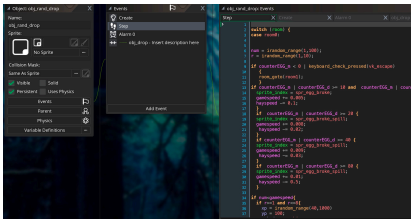


figure2: obj_drop

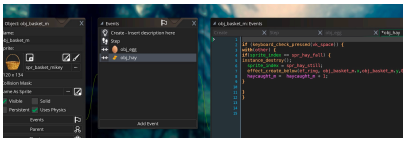


figure1: obj_basket_m

an item from obj_drop through the counterEGG variable, it will increase the games speed, change the amount of hay that is dropped and will also give us a visual on when to change specific sprites so we can subsequently communicate to the user that they are progressing well or losing the game.

2.collision objects - 1. characters
In this game, most sprites were assigned a fixed height, width, position, and spherical mask to help collision. This was created so that it would become easier to change the space they took up in the room. A collision instance between an item in free

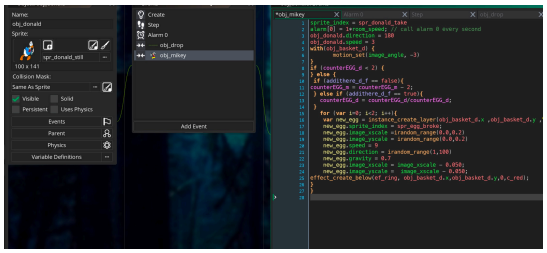
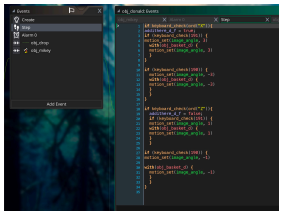


figure1: obj_donald - *collision example*

fall and a characters basket triggers a method so that the item in question will disappear. However, it will also gather data such as how many times it collected/encountered an item; if the player has pressed space to retrieve hay and will also change the eggs item count based on how much hay the player has caught. In figure 1 of Donald's collision, we can see that if you hit Mickey at a fast pace, you will lose half of the items in your basket and vice versa for the games opponent. Furthermore, broken eggs will be created and launched out of the characters basket to give a visual animation, so the player is aware.

 # CONTROLS



After the preparation of all of these functionalities, it was essential to develop controls for the characters so that they could move fast and slide throughout the room,

figure1: obj_m - *control example*

or in a slow incrementing fashion. Implementing this feature also led to the possibility of tracking the characters position and attaching other features to it; such as having the basket follow the characters x and y-axis or specific animations when colliding with the outer layers of the room.

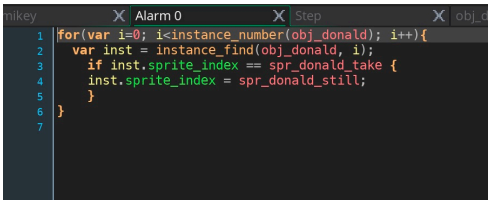


figure1: control and collision

When a collision happens between the two characters the alarm event of a character will look for a particular object inside of the room, cycling through all of the given objects instances, find and capture the data wanted and then change its desired sprite with the use of a "for loop".

To make the game feel more realistic, I added certain flourishes such as circle based animations when a character gets an egg. Additionally, the color of the expanding circle changes depending on whether the character catches an egg or collides with another character.

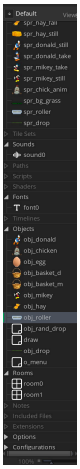


figure1: room and assets

 # ROOMS
The game contains two rooms that both have very different functions but are linked in their storyline. The main room holds the game itself and the other a menu to let you choose the difficulty, restart the game or exit the game altogether. Most of the objects, sprites, sounds, and rooms that are part of the game is displayed in figure 1 on the left side of the text. The game starts with a brand-new game in the main room so you can get playing right away. However, if you decide to exit the game by pressing the escape key on your keyboard, it will bring you to the menu room. The menu room consists of

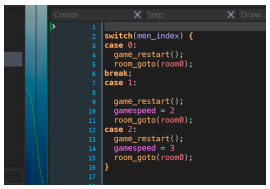


figure2: menu switch

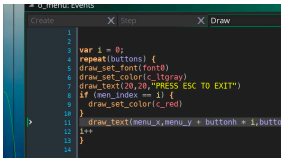


figure3: menu txt

an object with no sprite. I wanted to display a menu dynamically. The menu includes an array of strings that the up and down arrow of your keyboard can visually cycle through. Once a specific menu item has been chosen, a switch statement will trigger the particular case the user has selected. If they choose the first option, they will be guided to a new game, and the current game will restart. If they choose the second or third, the "gamespeed" global variable will be augmented at the start of game play, making it easy to manipulate the difficulty. This feature is shown in figure 2 In addition to these features, the game also

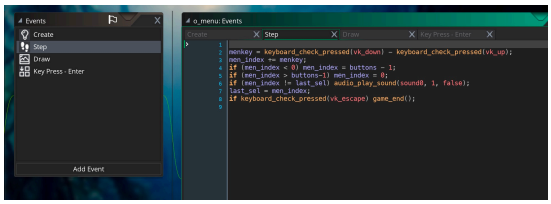


figure3: sound and selection

will also trigger a clicking sound on the up and down keystrokes when choosing from the menu.

DIALOG

Since both immersion and a storyline are essential to game play, it was crucial to build responsive dialog based on events that occurred throughout game play In figure 1, each string element will notify the player that they are running out of the following: 1. If one of the characters has caught 100 eggs, the game's overall gravity will lose all of its attachment and the game will show that the user has won the game.

figure1: dialog example

2. Since we made global variables based on the number of hay a user can catch, the player will be notified when it is necessary to gather more hay before catching more eggs.
3. If ten eggs have fallen out of the room or collided with the bottom room's layer, both players will be notified that they have ten more eggs to miss before they both lose the game.

CONCLUSION AND CHALLENGES

I found it interesting how most of the UI deals with an Object Oriented Mechanism. Throughout the project, I developed upon my ability to manipulate sprites, images, objects and their internal functions such as gravity, speed, direction… etc. Game Maker provides numerous tools to give you the opportunity to be as creative as you want to be. One aspect of the project I found particularly challenging was the interactive issues I ran into because I had been rushing to the next feature before finishing work on the previous one thus limiting the potential of what I could do each time. Once I realized that I could use most object events as triggers for different and specific behaviors in the game, I was able to work much faster and unlock new creative nuances that led me to build the game with its required specifications.

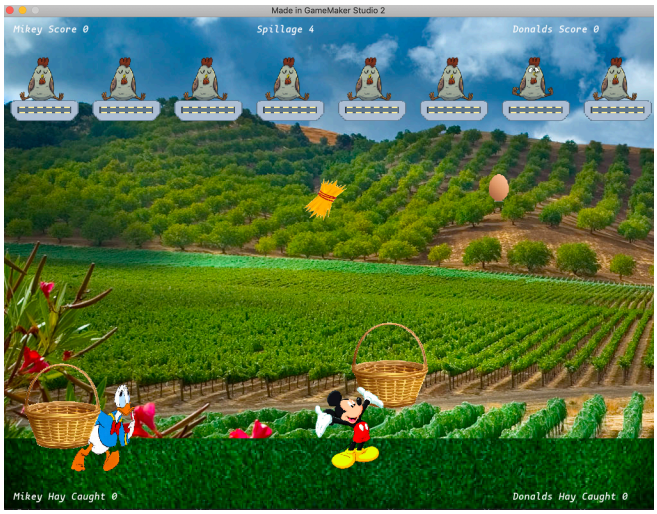


figure1: The Game