COMP 30540 Game Development

Egg Collector Report

Eoin Goslin (18204142)

Msc. in Computer Science

UCD School of Computer Science

University College Dublin

March 2019

Table of Contents

Introduction

This report outlines the steps and reasoning behind how I created the egg collector game. To start, I named my take of the game “Tropical Egg Collectors” as I remember playing Crash Bandicoot when I was younger and there were many tropical levels with great island-style music and I really wanted to try and emulate that style of game. This report outlines an overview of the game and some of the core game mechanics and logic before discussing extra embellishments that were added towards the end of this project.

Overview of the Game

1.1. Where the resources for this game were procured

All of the sprites for this game were retrieved from the open source website opengameart [1]. The game was created using GameMaker using their GML (Game Maker Language) to control the logic of the game. As I have never used GameMaker previously, most of the time was spent completing tutorials such as creating a Space rocks game [2] to learn GML and trying to learn how to use the GameMaker interface. This is the reasoning behind why not as much emphasis was placed on custom artwork.

For the sound effects, I created the menu song using an online tool for creating chiptune melodies called BeepBox [3]. The main soundtrack in the game as well as the sound effects such as the egg breaking were sourced from sound bible [4] another open source library. BeepBox was used to create the menu music, I wanted to create the feeling of playing an old arcade classic such as space invaders as the font chosen on the menu system looks similar to games made around the same time.

1.2. Menu and Help System

To create the menu system, an object with no sprite was created and placed in the first room the user enters when the game starts. A draw event was added to this object and in this event, a switch is used which takes as an argument the room parameter. Depending on what room the user is currently in, different messages are displayed.

```
case rm_start:
//Align Text in the centre
draw_set_halign(fa_center);

draw_text_transformed_color(
    room_width/2, 100, "Tropical Egg Collectors!",
    3,3,0, colour, colour, colour, colour, 1
);
//@ symbol in a string to type string over multiple lines
draw_text(room_width/2, 200,
@"Catch All the Eggs you Can!

Choose a Difficulty to Play By Entering the Number

1 - Normal Mode

2 - Extreme Mode: Are you eggsellent at the game? ;)

PRESS ENTER For Help

PRESS Esc To Quit
"
);

draw_set_halign(fa_left);
break;
```

Figure 1. Drawing Menu System

This object draws all the messages seen in the game so all the logic behind drawing messages is contained in a single event. This was chosen as debugging any issues with messages being displayed was a lot easier when I knew exactly what event and object was responsible for the action. After adding a case for when the user is in the help room, the result is as shown below.

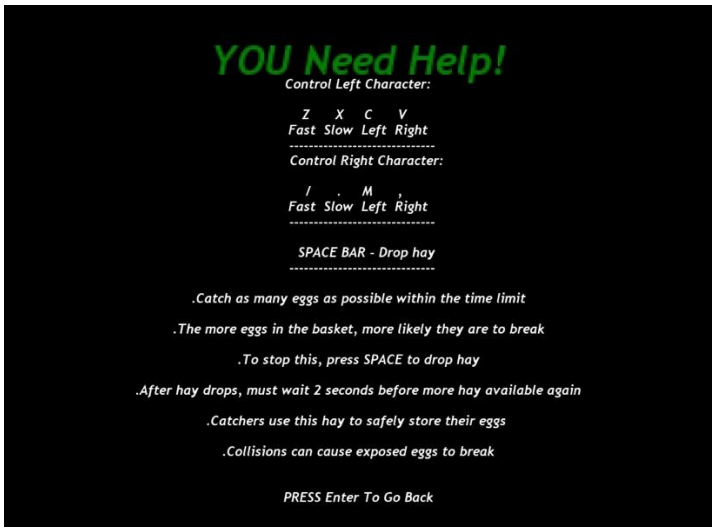

Figure 2. Help room

1.3. Alarms used

Alarms are a great feature in GML allowing for events to only occur after a specified amount of time such as when hay is able to be dropped again. These events are used throughout this game for actions such as creating/destroying objects, enabling/disabling actions and to keep track of in-game time.

First example of how I utilised the alarm feature is for destroying objects that no longer are required. The egg object in the game has a collision event with the land object so that when a collision is detected, the egg objects sprite is changed to that of a cracked fried egg. An alarm is then triggered so this object is destroyed after half a second. The reason an alarm is chosen here is because if the egg is destroyed upon collision the user would not be able to see any sprite change take place. Objects are also created based on timed events such as when a chicken may lay an egg in a random location. An alarm is called which will then change a random chicken's sprite to signal that this object is about to lay an egg and then the alarm is not called again until this event ends.

Another use for alarms was to ensure that hay could only be dropped after waiting two seconds since the last time it was dropped. This is done by using an alarm that runs every two seconds in the hay object. In the alarm, a Boolean is set to true if it is false. In the key press event, a check is carried out first checking if the Boolean is false or not before running an "instance_create_layer" function if two seconds has passed. After creating a falling hay instance, the Boolean is once again set to false and the alarm will run in another two seconds.

Lastly, alarms were used to keep track of the time in the game so the game finishes once a certain time limit is reached. In the game object, an alarm which deducts the global variable time limit by 1 and then sets a Boolean to false is run. In the draw event, if this Boolean is

false, it is then set to true and the alarm is set to go off in one second. This means every second the alarm is called; time limit is deducted and the variable displaying the time is changed.

1.4. Collisions Between Players

Depending on several factors including the difficulty chosen at the start, the speed players are moving and also how many eggs are in the basket, the outcome of any collision event will be different. If players collide with no eggs in their basket, then all that happens is the players are sent back the opposite direction, so they do not swap sides in the game room. If eggs are present in the basket, then a random number between either 1-20 (Normal) or 1-40 (Hard) is chosen. If that number is over 15, then the unprotected eggs in the basket will break. As can be seen, there is a greater chance of that number being over 15 on Hard but I wanted to make how fast the players were travelling at the point of impact have an effect on the outcome. The current speed of the player is added to check number meaning that if the player is going fast, 4 is added so if the players are going fast, there is more of a likelihood of the number being over the threshold of 15 and then the unprotected eggs breaking.

1.5. Collisions Between unprotected eggs and falling eggs

Similar to players colliding, a system which takes into several factors exists for this type of collision. Depending on if the difficulty is normal or hard and how many eggs are currently in the players basket, an egg landing on an unprotected egg in the basket may cause the player to lose all eggs in their basket. On normal mode, a number between 0-15 is chosen, 0-20 on hard, and then the number of eggs in the basket are added to this number so if a player has a high number of unprotected eggs then the chance of that number being over 15 is considerably increased. If the outcome number is over 15, then all egg objects are removed from the player's basket and their score is decreased.

```
//Now check if that player catching latest egg breaks other non-protected eggs
//On normal mode, there less of a chance that unprotected eggs will break
if !hard {

    var outcome = irandom_range(0,15);
} else {

    var outcome = irandom_range(0, 20);
}


//More eggs player has in basket, regardless of difficulty will cause eggs to break
outcome += num_eggs_in_basket_1;
if outcome >=15 and num_eggs_in_basket_1 > 1 {
    //Then eggs break
    num_eggs_in_basket_1 = 0;
    eggs_in_basket_1 = 0;
    x_offset_1 = 0;
    y_offset_1 = 40;


    //And destroy all the eggs that were in the basket

    var i;
    for (i = 0; i < instance_number(obj_basket_egg_left); i += 1)
    {
        var egg = instance_find(obj_basket_egg_left,i);
        with(egg) {
        totalPlayer1_eggs -=1;
        instance_destroy();
    }

  }
```

Figure 3. Egg breaking logic

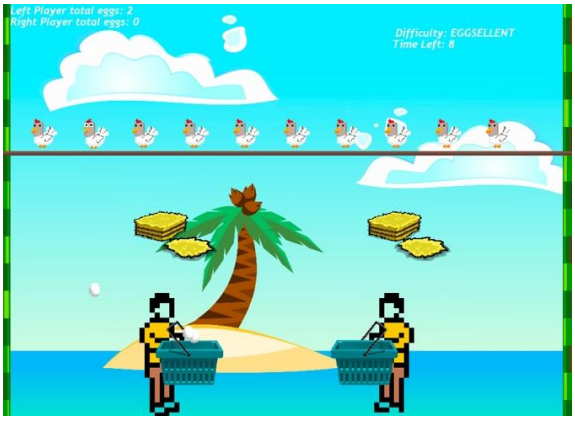

Figure 4. Workers rushing to collect hay

1.6. Movement of Characters

To ensure that players are never able to be still, “motion_set()”, an in-built GML method is used to point the players either to the right or left along the x-axis and depending on what key is pressed a speed is set as well. “Keyboard_check(ord())” is used to monitor which key the user presses and for characters such as a comma, their ASCII value is checked with just “keyboard_check()”. Figure 5 shows how depending on what key is pressed, and difficulty, a speed will be set to the given worker. On hard mode, workers move slower.

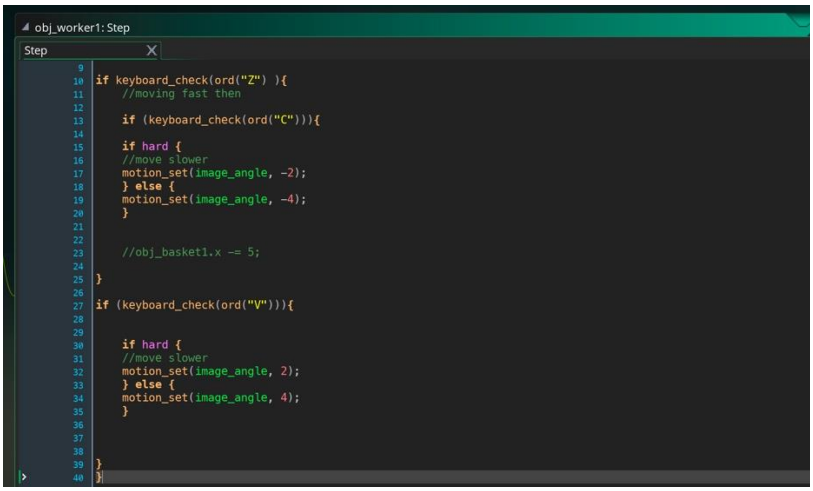

Figure 5. Worker moving logic

Extra Embellishments

2.1. Sound effects and Sprite changes

As discussed briefly earlier, sound effects have been added to the game. When eggs collide with a basket, hay drops, chickens lay eggs, or an egg hits the ground a sound is played. I wanted to add these as it gives more depth to the overall feel of the game if a player can hear events taking place. To play the audio, a GML method called “audio_play_sound” was utilised. First a check is done whether or not that the sound is playing already and if not, a sound file can be passed to this method and the sound is played so it is a relatively straight forward feature to add this feature.

As well as playing a sound when certain events take place, I wanted to ensure eggs falling to the ground change to appear as though they break. When hay drops also, a different sprite is generated as well as when a chicken lays an egg. To accomplish this, the "sprite_index" of a sprite can be accessed which can then be set to point to a different image file changing the look of that object.

```
//var randLocation = irandom_range(0, 5);
//chicken_ids is a list of all the chickens in the game
var chicken_laying_egg = chicken_ids[randLocation];
chicken_laying_egg.sprite_index = spr_chicken_laying_egg;
if !audio_is_playing(snd_chicken_laying_egg) {
    audio_play_sound(snd_chicken_laying_egg, 1, false);
    }
```

Figure 6. Changing sprite of a chicken laying an egg

Conclusion

I have made some games previously using JavaScript but after using GML I can really see the advantage. Almost any type of functionality for a 2D game is present with this language. Unfortunately, the trial license places a restriction on the number of objects that can be created which was the biggest challenge in making this game. This limited number did however force me to write clean code in the correct object and not use a vast number of objects to complete tasks that share similarities.

This game has been designed in a way to be a game that I would like to play, and I hope users who try it also enjoy the extra little features placed throughout. A lot was learned from creating this game and I look forward to creating many more games with GML.

References

1. OpenGameArt (2019) 2D - Complete Characters [online]. Available at: https://opengameart.org/content/2d-complete-characters (accessed 21 February 2019).

2. YouTube (2019) My First Game – Intro to GameMaker – Space Rocks [online]. Available at: https://www.youtube.com/watch?v=raGK_j1NVdE (accessed 21 February 2019).

3. BeepBox (2019) BeepBox [online]. Available at: https://www.beepbox.co/#6n31s0kbl00e03t7m0a7g0fj7i0r1o3210T0w1f1d1c0h0v0T0w1f1d1c0h0v0T0w1f1d1c0h0v0T2w1d1v0b4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4p16000000 (accessed 21 Feb. 19).

4. SoundBible (2019) Free Sound Effects [online]. Available at: http://soundbible.com/free-sound-effects-1.html (accessed 21 February 2019).