joshua hoban

u3174193

Game programming assignment report

# Problems faced:

A big problem I had throughout the assignment was trying not to repeat myself and fine more elegant solutions to save time coding, an example was turning on bounding boxes for my sprites. I placed a static bool in the parent ‘Sprite’ class of each entity in the game, this would allow for all bounding boxes to be turned on at once which save a lot of time in repeating the function.

Another problem was organizing the game logic of interactions between entities in the game to handle collisions. I chose to group similar entities (like my covid-19 spores) into lists so that they could be iterated easily. Then any interactions were than handled by the level state class.

Lag spikes occurred in my help screen which was a frustrating problem. It was cause by the draw routeing having to go through every call from the previous screen and then fulfill the draw routine of the help screen state, creating a very time-consuming process. The solution was to draw the previous state draw routine into a singular image by drawing it into a ‘RenderTarget2D’ object treating it like one whole texture. This saved lots of time for the draw routine of the help screen which improved performance greatly.

With all the different states and various classes requiring access to the games graphic device object or needing mouse input it became very troublesome and time consuming to continually create keyboard or mouse state variables when needing input or other objects. It was more efficient to have a static instance manager variable in some classes such as, my ‘MouseClass’ or “SoundLib’ were they could then be called from anywhere in the program without having to be initialized or put into class constructor parameters repeatedly.

Figuring out the volume level between all my sound effects and music was quite a challenge as whenever I lower the music, all the sound effects felt way too loud and therefore needed all be changed again, I just used some trial and error to figure out a nice balance that felt comfortable.

# Layering Strategy:

My strategy was to utilize the state manager to switch between states, record previous states for underdrawing when using menu states or applying game wide functions to all states by having it controlled in the origin class ‘Game1’, were functions like input checks to turn on bounding boxes or exiting the program could be applied to anywhere in the game and therefore didn’t need to be written multiple times in other levels or classes.

For drawing entities on the screen, I try to maintain each draw method of entities in their respective classes but each entity was a child of the parent ‘sprite’ class so all their animations, if there were any could be drawn easily and any conditions or parameters that are shared between all elements (like bounding boxes) could be updated efficiently. Any unique requirements such as the draw routine of bullets from the player class, or other entity specific functions would be containing in that specific entity class.

To draw overlaying menu screens where the previous state is displayed for example, in my pause or game over states. I added the menu state into to an array with the previous game state. Then in the draw routine I would call the previous state’s draw routine first, then continue with the current states draw routine. In the scenario of my help menu state there was a lot of lag due to the number of draw and update routines of the state, I instead drew the previous state into a ‘RenderTarget2D’ object were it was treated as a singular image or texture then draw the menu state.

# Story board:

A close up of text on a whiteboard

Description automatically generatedA picture containing text, whiteboard

Description automatically generated

Image on the left is the planning the layout of the help screen to try and give simple explanations in a minimalized style to save space on screen, while also working out the spacing relative to each object to make it easy to configure.

The bottom image is the planning on how elements would be displayed in levels and their function in game levels.

Many changes were made during the assignment the toilet paper was originally going to give a shield, but I thought it be more fun to restore health and destroy all enemies on screen. however, I stayed mostly to the initial plan.

# Reference:

Music:

Cuphead OST - 07 - Botanic Panic,

<https://www.youtube.com/watch?v=kVLZ9rNRHsU&list=PLTSIqhWP1gKVcDruitj0nq1mN3B--PUPh&index=7>

Sound effects:

<https://freesound.org/>

Sprite sheet for buttons:

<https://www.kenney.nl/assets/ui-pack-rpg-expansion>

Font Style:

<https://www.kenney.nl/assets/kenney-fonts>

Background Image sources

<https://www.corporatewellnessmagazine.com/article/addiction-to-better-health-and-wellness>

<https://www.forbes.com/sites/linhanhcat/2020/01/29/how-high-in-the-sky-can-microbes-survive/#2fdb6dd6734c>

<https://www.premiumbeat.com/blog/color-correcting-tips-pink-sky/>

<https://en.wikipedia.org/wiki/Moffett_Federal_Airfield#/media/File:Kluft-photo-Moffett-Federal-Airfield-Oct-2008-Img_1911.jpg>

<https://www.steelmasterusa.com/residential-buildings/aircraft-hangars/>

Code sources:

State Manager Class:

<https://rareelementgames.wordpress.com/2017/04/21/game-state-management/>

Sprite class: (Batholith Entertainment):

<https://www.youtube.com/watch?v=Qx4Re5NTmq0>