PAVEMENT FIGHTER

Console Development Documentation

AE1 – CGP500

Fate Show – Q14642808

Contents

Please note not all classes have their own page, some may just be found inside the testing section and vice versa. This will be mentioned in more detail later in this document.

Design section

Original Game Description

The player plays as a slightly overweight drunk man in a slightly too-tightly fitting top, who must fight off the rowdy hooligans who stand in his way to the best bookie in town.

Original Details

Enemy attack mechanic:

- Every 0.5 sec there's a 50% chance that the enemy will attack (punch/kick).
- If the player punches, there's a 30% chance that the enemy will Duck.
- If the player kicks, there's a 30% chance that the enemy will Block.
- These probabilities will increase depending on the difficulty of the enemy.

Original Level design:

Initially my aim is to create one level that has two smaller enemies and one boss enemy at the end. The player will start on the left side of the screen on a pavement, and they will progress by walking towards the right direction where they will come across the enemies. At the end of the level on the right of the screen will be the target destination – the bookies.

If I am able to, I will create more levels by using the same framework used for the first level. This will be through the use of enemy parent classes that all enemies will inherit from, loading level tile maps through an external file, and ensuring that my code is modular with the use of Object-Oriented Programming.x

Design section

Final Game Description

The player plays as a slightly overweight drunk man in a slightly too-tightly fitting top, who must fight off the rowdy festival-going chad who stands in his way.

Final Details

Final Level design:

Once I went back to properly observe the original 1987 Street Fighter game, I changed my design slightly to match it better. There is now only one level, and the player can move left and right but the screen doesn't scroll. All the attack mechanics are the same. There are no longer any sobriety pick-ups.

If there are errors, I've screenshotted my settings below. Though your file path may be different, this SDL folder is included within my submitted game files.

NOTE: MAKE SURE YOUR COMPILER IS IN 'Debug' MODE x86!

Flowcharts and Pseudocode

Initial Higher-level flowchart

Flowcharts

Final Higher-level flowchart after adjustment

Player attack pseudocode

Player block pseudocode

Flowcharts and Pseudocode

Enemy attack pseudocode

Enemy block pseudocode

UML Diagram

Original UML diagram

UML Diagram

Final UML diagram after adjustments

Character Design

Initial Sketches

I've gone for a blocky style of art as it's an amusing style of pixel art that I enjoy making, in addition, the assets don't
much detail when scaled up or down. I initially was going to have pick-ups, but my final game didn't use them

Player: Enemy: Pick-ups:

Pixel Art

Player: Enemy:

Project documentation

I have drawn inspiration from this YouTube channel for some of my class design https://www.youtube.com/playlist?list=PLhfAbcv9cehhkG7ZQK0nfIGJC_C-wSLrx .

Player

The player class is inherited from Entity and holds the player's input, alongside anything else related to the player.

Problem 1: When I first created the Player class the texture wasn't rendering to the screen, despite the console loading the image and calling all the relevant functions needed (Player object is named Keith).

I then released that I had overridden the Update() function inside player, so that the texture wasn't being rendered to any source and destination rectangles, so when I commented out the player's update, it worked. I will just need to call UpdateInput() in the Game class.

Enemy

The Enemy class is inherited from Entity and holds the attack functions as well as a random number generator that helps to make enemy attacks less predictable. Refference: https://stackoverflow.com/questions/34490599/c11-how-to-set-seed-using-random

Here RandNum() just prints out the distribution to check it's generating random numbers, which it does successfully.

I had wanted to pass the UpdateAttack() function into the RunTimer() function using the #include<functional> library (refference: How do you pass a function as a parameter in C? - Stack Overflow) but, when I did that, the function ran every frame rather than during my pre-set intervals. However, the timer wasn't broken.

So, I changed RunTimer to return true if the timer interval was complete and implemented that so that it worked. Printed 8 "ATTACK"s in 16 seconds at a time interval of 2 seconds.

Entity

Entity contains functionality that's used in all the characters needed in my game such as SetTexture(), RenderEntity() and SetPos().

The collision code that all characters use to check their attacks have collided.

The first issue I came across is a memory leak when I tried implementing the sprite sheet. I was loading the sprite sheet every frame which consumed huge amounts of memory.

So, I tried to create a vector of frames and loaded the textures in only when I created each object. But despite the images loading in successfully, they weren't displaying. I was under the belief that the issue was in AddFrame().

So I went back to the original way of how I loaded in the sprite sheet, and you can read about that in the testing section below in test 6.

Input

Input class takes in user input through the controller. The input pointer mp_input is created in Player, as it made more sense to me as the player class was using the Input functionality most.

A problem I faced is that when the right button was pressed, the player continued going right without stopping.

And I found that the button was being pressed down but not up. This was because somehow a second switch statement had wrapped itself around the key up switch.

After removing that, the player moved as intended.

Scene

Scene shows the background image. But it wasn't loading originally properly. This was due to int i being reset inside the while loop.

After some adjustment it then worked.

FontManager

FontManager handles all UI text-oriented tasks. It uses the SDL2_ttf font library functionalities to create an overlay that is passed to the renderer.

Init() is where it's initialised, returning an error if the TTF can't be retrieved. It's called in its constructor. DrawToScreen() displays the text to the renderer.

SetText() takes in a string and outputs it to the window, each frame.

To output the player and enemy's sobriety, I created a string stream in Game() which formats the string correctly.

Audio

Audio class handles the music and sound effects. The background music can be found here: https://freesound.org/s/251461/.

The attack SFX sound can be found here: https://freesound.org/s/404766/.

Sound effect implemented in Punch() and Kick().

Music implemented in Game() constructor.

PointerManager

PointerManager() is a static utility class that I can use to delete pointers. It is better practice to use smart pointers however I wanted to learn how to create static utility classes and I also wanted to learn how to use a template – and where to practically apply them so that they're actually useful. In addition, it was inefficient to always type out:

Below is the PointerManager():

And after including its header file, it can be used like so:

Here are the resources I used to learn about templates: https://www.youtube.com/watch?v=I-hZkUa9mIs&t=430s and https://stackoverflow.com/questions/41225738/c-template-function-to-delete-and-reset-pointers .

It was working but then this happened:

Then after some scrutiny I realised that I was deleting some pointers that I hadn't called new on, like the texture pointers below.

So, I removed the deletes where I didn't need them, and it worked and exited with code 0.

Timer

Timer() class is used for the enemy attack timings. I used these websites as refference:

- https://www.fluentcpp.com/2018/12/28/timer-cpp/
- http://lazyfoo.net/tutorials/SDL/22_timing/index.php
- https://stackoverflow.com/questions/56228900/how-can-i-implement-a-timer-using-sdl2-libraries-in-c-making-lunar-lander-game
- https://www.libsdl.org/release/SDL-1.2.15/docs/html/time.html
- https://www.libsdl.org/release/SDL-1.2.15/docs/html/sdladdtimer.html
- https://www.fluentcpp.com/2018/12/28/timer-cpp/

This is the test function to see if it would print every set time interval. Here I've set it to 2000 milliseconds.

I then added the code for limiting frame rate into the Timer class, so that it was all in one place.

But then I got this error:

I think that this was due to my Game's frameTimer and Enemy's attackTimer both being pointers, pointing to the Timer function. So, the program got confused with what the Timer should've been doing. So, I just changed the enemy's timer to be an object rather than a pointer and it was fixed.

MessageHandler

MessageHandler class stores all the important messages that require the user's attentions.

Such as if a controller is disconnected:

This one gives instructions on how to exit the game. This button enters the game when clicked.

Or some notify the player if they've won or lost. These buttons successfully exit the game.

Testing

Table

Testing

Evidence and Explanation

Testing

Evidence and Explanation

Testing

Evidence and Explanation

Testing

Evidence and Explanation

Testing

Evidence and Explanation

Testing

Evidence and Explanation

Testing

Evidence and Explanation

\

Testing

Evidence and Explanation

Testing

Evidence and Explanation

Conclusion

What went well:

I have managed to implement a lot of features and have even tried to improve the user experience through finding the appropriate music, tried to maintain an art style that reflected the humour, and even added a highlight to the kick attack to show that it's a higher charged attack. I also originally had a more square/pixelated font that I later changed so that the text was more obvious to the player. I've documented that I've spend roughly 67hrs on this project including the research, testing and documentation and though this is a lot, I believe that I've learned a lot and have improved dramatically. I now feel more confident in SDL2 whereas I wasn't before the start if this project. I feel like I've created a game that, though may not be very good, is playable from start to end.

Even better if:

Unfortunately, I wasn't able to implement the block and duck mechanics other than the player animations. However, I have the pseudocode for it and have set up my functions for the addition of these features and I feel that I can implement these, should I have spent longer on this project. Despite this, I do feel like the game is playable without the addition of these, but it would've improved the game play.

Also, despite my efforts to limit frame rate, it is inconsistent due to the memory leak that occurred near the end of this project and I wasn't able to work out where this was coming from. But my game doesn't crash so as memory leaks go, it's not the worst one I've had.