Alexey Rirak

Documentation

The final game that I was able to come up with is a remake of the first level of the NES version of Super Mario Brothers. Most of the features of the original game are implemented with a few exceptions. The features I did not implement are the Koopas (turtles), the fire Mario mode, two player mode and check points. When I first began developing the game I wanted to also implement a high score system for the game, something which was not present in the original, but unfortunately I did not have enough time to do this. Mapping out just the first level proved to be an extremely long task. Having a full sketch of the level really helped (this is detailed in the first update). The final level array was over 200 tiles in length and 15 tiles in height.

The following is a list of (most of) the implemented features along with an explanation of how they were implemented.

Mario Character
The main playable character is Mario. The whole game is centered on this character's movement. The character is almost always fixed at the center of the screen (except the very beginning and very end). The background ("the world") moves in order to create the effect of the character moving. The playable character is implemented as its own class with the ability to figure out its movement based of keys being pressed. The character is controlled by using the W, A, S and D keys. This character is a highly modified version of the character which was demonstrated in class by Professor Kapp. The character has been modified so that he cannot jump through solid objects and so that he keeps track of his own stats. Mario keeps track of the score he has earned so far and the amount of coins he has collected. As well as weather he is in super mode. When in super mode, the character grows to double the size and is able to run into enemies without dying (though this switches him back to normal mode). Additionally, while is super mode the character is able to break bricks by hitting them with his head.

Various Blocks
The game implements a few different kinds of blocks. The graphics were all taken from tile sheets

and cut up into small sections for use in this game. The blocks are divided into two general classes solid and not solid. Simply put not solid blocks can be passed through while solid blocks cannot. This was decently easy to implement as we have done a basic version of this in class. The difficult piece was the fact that there are special blocks. While non solid blocks don't have any special behavior some solid blocks do. Here I had to implement "?" blocks and brick blocks. "?" blocks produce either a coin or an item. There are some blocks which produce a coin and some blocks which produce a mushroom. Each of these is implemented as its own object which keeps track of its own position. Coins simply fly up and come back down before disappearing and increasing the player's score. Mushrooms move around the level following the laws of gravity until they are either caught by the player or they move off the screen. The most difficult kind of special block is the brick blocks. These blocks don't do anything special when the player is normal but when the player goes to super mode these blocks become breakable. It took me a while to come up with a way to do this but eventually I ended up creating an object which would represent each piece of the block. When hit the block breaks into four separate pieces which go flying in different directions. Each piece keeps track of its own location and is removed from the level once it hits the ground.

Goombas
Goombas are the little enemies you see wandering around in Mario games. They are typically the easiest to defeat but implementing them is not as easy. The basic behavior is that they walk around back and forth switching direction every time they run into a solid object. If a Goomba runs into the player the player dies but if a player jumps on top of a Goomba the Goomba dies. This proved to be a difficult distinction to make because in class we've only discussed general collisions but eventually I was able to find a way to do it. The Goomba keeps track of collisions and if a collision occurs it checks if the player had any "jump power". If they did they were in the air so it means the player probably jumped on top of the Goomba. Otherwise the Goomba ran into the player thus the player died. While this does produce some false positives sometimes overall it proved to be a pretty reliable method.

Management Classes
The other classes I implemented are management classes. These are classes that manage some specific function of the game. Theoretically these should be implemented using the Singleton pattern but I figured this might cause problems in JavaScript mode and wasn't sure if processing

supported the Singleton pattern in general. These classes include the soundDB class, levelDB class and menuDB class. The soundDB class manages all the sounds which are used within the game. This makes it easy to control and alter any sounds if necessary. The levelDB class contains all the information about the specific level being played. This includes the tile mapping of the level, enemy locations, game time, solid and non-solid tiles as well as all the helper function which provide information about the world (such as the tileIsSpecial function and functions to animate tiles). It also implements the animation at the end of the game where Mario goes into his castle. The menuDB class implements all the menus used within the game. This includes the start screen, death screen, game over screen, winning screen, time up screen, and the HUD which is displayed while the level is played and lists the score, time and coins collected.

Extras
To assist in the development process I implemented a debug mode which displays useful information about the character to the screen and shows collision checkers for moving objects. There is also a location mapping mode which displays the location of the character so that locations can be found for placement of enemies and items.

Challenges
There were many challenges which I encountered during the development of this game. Figuring out how to go about implementing many of the features was often an issue. One of the early issues I faced was that independent object would move to quick when the player moved because they weren’t attached to the level. This meant that the level moved behind them and created the illusion that they were moving extremely quickly. The solution was attaching them to the world so that they would move with it. I also had an issue with these object passing through solid tiles which took me a while to fix. The breaking of the brick block took a huge amount of time and required me to simulate physics which is difficult to do without a physics engine. Throughout the whole process maintaining JavaScript compatibility was a huge issue. There are many things that work well in Java but are not carried over to JavaScript correctly. The game does work decently well in JavaScript mode but there are still some glitches that exist in JavaScript mode which are not present when running the game in Java Mode.