CMP105 Coursework Report: *Dungeon Dodger*

Student Name

1901881 – Joseph Roper

Git Username: 1901881

Introduction

I have made a top down dungeon crawler type game. The aim of the game is to get to the end of the dungeon while only dodging enemies and defeating them without attacking. I took inspiration from the Binding of Isaac and Enter the Gungeon. The title of the game is Dungeon Dodger and it is about a cocky brat who's always been good at dodging and to prove her skill she enters a dungeon and decides to only dodge. The aim of the game is to get to the end of the level without dying. In dungeon dodger there is collision between the enemy and the player which decreases the players health and the enemy slimes can collide with the grids on the floor and die. The player can also collide with the entrances to rooms to teleport to the next room. Dungeon dodger contains multiple sprites such as the player, slimes, health bar and the grid. For the enemies and the player, I used my own created sprite sheets to animate them. The game I created contains numerous screens such as the menu screen which is displayed at the start of the game where the user can click to access the level, controls and story screen. From the level the player can press escape to access the pause screen and if the player dies, they go to the death screen, lastly if the players gets to the end of the level, they will access the win screen.

My game plays different music depending on what screen you're on and plays a sound when the player is hit. Dungeon Dodger has a unique mechanic and that is with the slimes, there are 3 slimes the blue one which is the basic slime with average speed and dies in one hit, the yellow slime dies in one hit as well but is a lot faster and finally the red slime which is bigger and slower than the other two slimes and takes two hits to die and after the first hit the slime gets smaller.

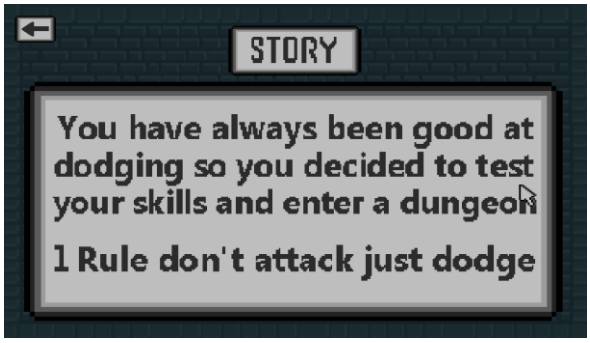
Controls

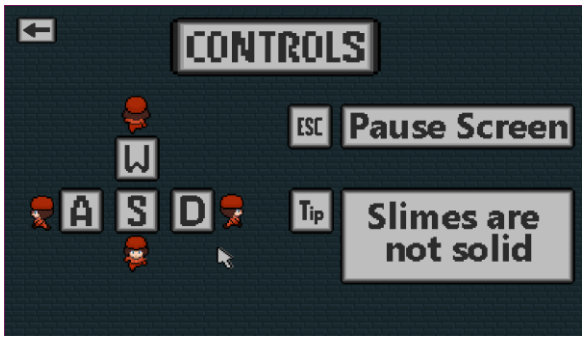
The player can move the main character up by pressing W on the keyboard, down by pressing S on the keyboard, left by pressing A on the keyboard and right by pressing D on the keyboard. The player will also be able to press the escape key to access the pause menu. While in the other game screens the player will be able to press buttons with left mouse click and access other game screens or exit the game.

Game Screens

Dungeon dodger has multiple screens the; title screen, level screen, pause screen, story screen, controls screen, win screen and the death screen. The title screen is displayed at the start of the game and from it you can use the mouse to click on the buttons to got to the controls, story, and level screen or to also exit out of the game. The level screen is where the user plays the game and from it you can press escape to go to the pause screen, have the player die to go to the death screen or get to the end of the dungeon to get to the win screen. From the pause screen you can access the story, controls and title screen as well as closing the game by clicking the buttons, you can also go back to the game by clicking the back button. While on the story and controls screens you can click

the back button to go back to the previous screen. From the win and death screen you have the option of going back to the title screen with the restart button or exiting the application.



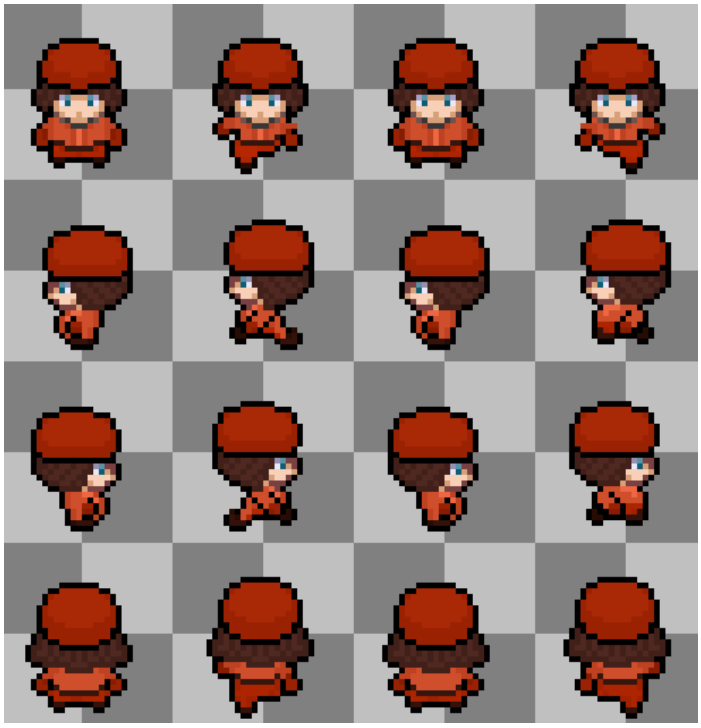


Input

Within my game the player can use their mouse while in the menu screens to click buttons to switch between screens. While in the level the user can press the escape key to access the pause menu and use WASD to move the player around. When the player accesses a different game state the corresponding class is loaded up which changes the texture displayed and adds the clickable buttons. The buttons are created in the constructer of the class and drawn in the classes render function. The handleInput function makes the buttons clickable it checks to see if the cursor is within the collision box of the button and if it is it sets the gamestate to the that of the text on the button.

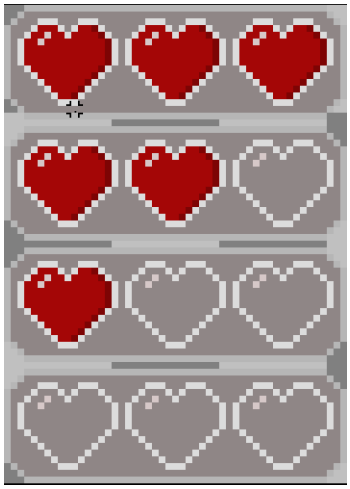
Sprite Work

All the games 2d pixel art was designed by me, the sprites animation was done by creating a sprite sheet which contained every frame of animation it was then set as the sprites texture in level and set up in the sprites class constructor. In the sprites class header file, an animation class object is created for the specified animation. Then in the constructor I add the different frames of the animation of the object using a function and mapping out the correct coordinates of the frame from the sprite sheet. I then animate the sprite in the classes update function and change the sprites animation in the handle input function depending on what the sprite is meant to be doing. For example, if the user is making the character go left the current animation being played will be swapped to the left animation.



For the slimes animation I made a lucid 6 frame sprite sheet which when played makes the slime look like its jumping, the sprite animation is omnidirectional which saves the code from having unwarranted animations for different directions. The seamless animation is played on loop when the sprite is moving and is reset and stopped when the sprite is still. I believe this enemy movement animations makes the game look more complete and more immersive. The animation works the same as the main characters except it just plays and loops the same animation while the slime is moving.

In the game the player has a health bar in the top left of the screen which decreases as the player gets hit. The health bar is animated with 4 frames ranging from 3-0 hearts full to tell which frame the animation should be on I implemented two integer type variables healthMax and health. The variable healthMax will always stay at 3 whereas the health variable decreases by one every time the player gets hit by a slime in the level's collision response. I made functions to retrieve and alter these two variables so I could use the private data in the level and hearts class. In the heart's classes update function I set the current frame to be the healthMax minus the current players health. This sets the sprite to the correct frame.



Collision Detection and Response

Dungeon dodger uses the sprites position and the rooms position in the checkBoundaries function in level to keep the mc and the sprites within the boundaries I set of the current room. I check the sprites x and y position against the rooms centre plus or minus a set number to keep them in the room. If the sprite is within the boundaries they can move however if they get out of the boundaries they are teleported back into the room.

```
//Level Boundaries
//the boundaries are updated based on what room the player or enemies are in
if (currentObject->getPosition().x <= tileObject.getRoom(currentObject->getCurrentRoom()).getCenter().x - (657))
{
    currentObject->setPosition(currentObject->getPosition().x + 1, currentObject->getPosition().y);
    currentObject->setVelocity(0, 0);
}
else if (currentObject->getPosition().x >= tileObject.getRoom(currentObject->getCurrentRoom()).getCenter().x + (532 - scale))
{
    currentObject->setPosition(currentObject->getPosition().x - 1, currentObject->getPosition().y);
    currentObject->setVelocity(0, 0);
}
else if (currentObject->getPosition().y <= tileObject.getRoom(currentObject->getCurrentRoom()).getCenter().y - (362 + scale))
{
    currentObject->setPosition(currentObject->getPosition().x, currentObject->getPosition().y + 1);
    currentObject->setVelocity(0, 0);
}
else if (currentObject->getPosition().y >= tileObject.getRoom(currentObject->getCurrentRoom()).getCenter().y + (242 - scale))
{
    currentObject->setPosition(currentObject->getPosition().x, currentObject->getPosition().y - 1);
    currentObject->setVelocity(0, 0);
```

All the other collision used in my program is with set collision boxes. In my sprites class I set a collision box for every frame of animation so that the collision box is always the right size and covers the sprite. If the player is within any of the slime's collision boxes the hit sound effect is played, the players invincibility frames are added so that the player can't get hit 3 times repeatedly and die straight away and the players health is lowered by 1. If any of the slimes are within the collision box of the grid object, depending on what room they are in they are teleported to a different room as well as getting invincibility frames added.

```
blueSlime->addInvis(5);//stops the slime from getting hit by the grid again
//too save data the slimes dont die they just get teleported to a different room
if (blueSlime->getCurrentRoom() == 6)
{
    blueSlime->setCurrentRoom(1);
    blueSlime->setPosition(tileObject.getRoom(1).getCenter().x , tileObject.getRoom(1).getCenter().y );
}
if (blueSlime->getCurrentRoom() == 5)
{
    blueSlime->setCurrentRoom(6);
    blueSlime->setPosition(tileObject.getRoom(6).getCenter().x , tileObject.getRoom(6).getCenter().y );
}
if (blueSlime->getCurrentRoom() == 4)
{
    blueSlime->setCurrentRoom(5);
    blueSlime->setPosition(tileObject.getRoom(5).getCenter().x , tileObject.getRoom(5).getCenter().y );
}
if (blueSlime->getCurrentRoom() == 1)
{
    blueSlime->setCurrentRoom(4);
    blueSlime->setPosition(tileObject.getRoom(4).getCenter().x , tileObject.getRoom(4).getCenter().y );
}
```

```
//checks to see if the player has been hit by a slime
if (((Collision::checkBoundingBox(mc, blueSlime))
    || (Collision::checkBoundingBox(mc, yellowSlime))
    || (Collision::checkBoundingBox(mc, redSlime)))
    && !mc->getInvis())
{
    audioManager->playSoundbyName("hitSound");//plays thi
    mc->addInvis(0.5);

    int x = mc->getHealth() - 1;//when hit decreases heal
    mc->setHealth(x);

    if (mc->getHealth() <= 0)//when the players health hi
    {
        mc->setAlive(false);//this stops updating the pla
        gameState->setCurrentState(State::DEATH);
    }
}
```

Audio

In my game I have two music tracks the first one is played while the player is in menus, I do this by in the menu states handle input I stop all the music and then play the first track when the button is pressed before the program goes to the next state. The second track is played when the level is loaded and plays when the player is in the pause, win and death screen using the same method that was used to play the first one. I set it so both the music tracks loop.

```
audioManager->addMusic("sfx/On_My_Way.wav", "menuMusic");
audioManager->playMusicbyName("menuMusic");//plays the ad
```

```
audioManager->addMusic("sfx/A_Journey_Awaits.ogg", "levelMusic");
audioManager->addSound("sfx/BABABOOEY.wav", "hitSound");//adds a
```

```
if (Collision::checkBoundingBox(storyB, cursor) && (input->isMouseLDown()))
{
    audioManager->stopAllMusic();
    audioManager->playMusicbyName("menuMusic");//plays the added music
    gameState->setCurrentState(State::STORY);
}
```

I also have a sit sound effect that plays when the player is hit by a slime. I do this in the levels handleCollision function and if the players collision box hits any of the slimes it plays the sound file.

```
//checks to see if the player has been hit by a slime
if (((Collision::checkBoundingBox(mc, blueSlime))
    || (Collision::checkBoundingBox(mc, yellowSlime))
    || (Collision::checkBoundingBox(mc, redSlime)))
    && !mc->getInvis())
{
    audioManager->playSoundbyName("hitSound");//plays th
    mc->addInvis(0.5);
```

Game Logic and Unique Mechanics

In my game the slimes have player tracking. Meaning that they can get the direction of the player and start moving towards them. I do this by setting the target to be equal to the players position. I then have direction set to target minus the slime position the direction is then normalised and used to calculate the slimes velocity which is used to make the slime move. Another unique mechanic is that the player can't attack so they have to lore the slimes over a sewer grid to make the slimes fall through it using the slimes player tracking mechanic too full effect.

```
//Calculating target
//target is used to calculate the direction the slime will move in
target = sf::Vector2f(mc->getPosition().x, mc->getPosition().y);

//calculate direction and move
direction = target - getPosition();
direction = Vector::normalise(direction);
velocity += (direction * acceleration) * dt;

move(velocity * speed * dt);

// if object is close enough to target
if (Vector::magnitude(target - getPosition()) < 10.f) {
    setPosition(target);
}
```

Another unique mechanic is that the red slime within the game has two lives, the slime starts off as big and once hit decreases in size. I do this by setting the size of the red slime to be twice as big as the other slimes and the collision boxes for each frame of animation to be bigger as well. Then once the slimes collision is activated and the slime is hit the slimes size is decreased along with the collision boxes.

```
if (Collision::checkBoundingBox(redSlime, grid1) && !mc->getInvis())
{
    redSlime->addInvis(10);//stops the slime from getting hit by the
    //too save data the slimes dont die they just get teleported to a
    double scale = 0.5;
    redSlime->setSize(sf::Vector2f(254 * scale, 254 * scale));//if th
```

Conclusion

I believe the art and animation for my game went well as I have never made art before letting alone game art so for this coursework, I decided to set myself a challenge and make art and animations for the entire game. I found this difficult and time consuming at first but began to get used to pixel-based artwork and how to animate characters by creating midframes in between the desired action. I also believe that the different game states I created transfer over smoothly with the button functionality implemented.

I found it quite difficult to make the in-game tile map as I made a lot of small errors which were difficult to find and debug such as missing a number out on the tile map. I struggled with getting the correct screen size for the game which would include the whole dungeon room I created, to fix this it took a lot of trial and error to decide which size looked best and didn't cut anything out. What made the coding process of the game difficult was not knowing which functions already existed for what I wanted to do and which ones I would have to create to suit my purpose and how I would code it.

In the future I would like to add more enemies into this game for example a skeleton that throws its limbs at the player until it can't move but couldn't implement because of time constraints. Secondly, I would increase the map size and add more levels with different art. Thirdly, I would make more sprite animations too make the game look more polished for example a death animation for the player and for the slimes where they would swirl down the grids. I would also add button animations to make the buttons light up in the cursor was over them.

To improve my development process when starting a new game project, I would write out every feature I want and need for the game and select the features that are necessary for a working game demo and focus on them first for example making the enemies and player then implementing their movement. Then I would write down how I think I would implement every feature and go through them one by one making sure they work properly before starting another. For the art side of the game I would make a basic tile map such as different coloured squares and start coding with them and then slowly make the actual game art as the project goes on and swap the tile maps out. As for this project I regrettably didn't start on tile mapping before I had completed game art of the dungeon slowing down my work process. Although I believe all of these were necessary mistakes which will help me streamline my future projects.

References

Music used in menu screens:

DeltaBreaker, 2017. *On My Way*. [online] OpenGameArt.org Available at:

< https://opengameart.org/content/on-my-way-8-bit-loop>

Music used in the level and other screens:

pbondoer, 2014. *A Journey Awaits!*. [online] OpenGameArt.org Available at:

< https://opengameart.org/content/a-journey-awaits>

Sound used for when the player is hit:

The G-Forse Experience, 2014. *BABABOOEY Sound Effect*. [online] OpenGameArt.org Available at:

< https://www.youtube.com/watch?v=U_cPir6MwLM>