Individual Reflection

Game Programming C++  
SDL game project: Bullet Hell

Team 3

Fredrik Hanses  
  
Controller support system: InputHandler.cpp + .h  
I imported the core system from a tutorial that I later expanded. It works seamlessly with the keyboard input system and works in menus and in gameplay. The game supports up to two players and they can choose to play with a controller or with the keyboard. They can even mix input systems.   
  
This system still needs a bit of polishing though. Joystick drift often occurs, and buttons sometimes have a hard time register that it has been released. In a proper game, controllers should be able to get unplugged and plugged in again and work. In our game they do not. There is no key/button reassigning either.   
  
Playable characters: Player.cpp + .h, Player2.cpp + .h  
During the project I have worked extensively with the player characters. It was a natural choice because I started with the input system. The impact and explosion effects were ported from the box character.   
  
The second player is essentially a clone of the first but with some recoloured art assets and its own control scheme. There is a lot of code duplication between them and this should be done smarter. It was a prototype idea that evolved and never got changed.   
  
Box character: Box.cpp + .h, BoxPiece.cpp + .h  
I created the box as our first enemy prototype, but it eventually evolved and became our fully functionally but only adversary. Over the course of the project I added new effects to the box character. These effects were the impact effect and explosion death animation. Both of which I ported to the players except of the flying broken pieces.  
  
The boxes really need more variation, both in appearance but also in movement patterns and attacks. We originally wanted to have several unique bosses but that work never got started. Maybe that was for the better because of how bland even the final enemies become.   
  
High score system: ScoreManager.cpp + .h  
When I started working on the high score system it already saved and displayed the best high score. My task was to save it between playthroughs. I then developed the reading and writing systems that saves the high score to a text file. I later adjusted it to accommodate the ability to save the seven best high scores that shows up on the credits screen.   
  
This system can improve a lot. It can only read and write high scores. To read something else requires a lot of adjustments. It writes a small amount of data to several files instead of writing it all to one file. I implemented it this way because it was derived from a system having just one high score and I had a lot of problems just getting that to work. I did not want or had the time to reimplement the system. For our game though, it works perfectly.

Gameplay state: PlayScreen.cpp + .h, Level.cpp +.h  
Working with the input system and the players made me test the gameplay a lot. Therefore, I worked with the gameplay state and added functionality primarily related to the players.   
  
General bug fixing  
Bugs that I ironed out was mostly around code that I had originally written. But I also fixed some bugs related to the user interface or the gameplay loop.  
  
Non-programming tasks  
I searched for and selected a lot of the backgrounds that we use for our different levels. I also choose a lot of the music in the game. The same goes for the sound and visual effects. I also recoloured all the sprites that needed it.   
  
I think C++ have been a challenge for most of the class. I feel that during the project I have learned to use it almost as good as C#. It is more cumbersome sure, but probably a more powerful tool.   
  
At first SDL felt very archaic and rigid. But after following a tutorial and creating my own engine before the project, following the lectures and co-creating two different engines and then working on a project with yet another SDL engine I can safely say that it has grown on me. But it is still a wonderous thing that other more sophisticated game engines exists that do this heavy work.   
  
The communication in our group was a challenge. One of our group members family situation required him to work at odd hours while I tried to be at school and work during office hours. The other one got seriously ill and could not attend for a while. But we used Discord extensively and that have worked for us. We have had some meetings face to face but not many and they were mostly in the beginning of the project.  
  
When I got the controller system to work it became so much more fun to play the game and that made us choose to have a two-player co-op mode. I feel that that was a very good design choice because it also made this simple little game to become even more fun to play. Another thing I was very pleased with was when I finally got the high score system to properly save and load high scores. It took a very long time and had a lot of bugs along the way, but it worked out in the end. I also think that Git as a version control system has worked very well for us during the project. There were not many issues and those that arose we could solve quickly. It has been a great tool as it should.