**System Sequence Diagrams**

1. **Dash –**



1. **Checkpoint –**



1. **Pause Menu –**



1. **Title Screen –**



**Class Diagram**



**Designing Explanation**

For the class diagram, every object is associated with the player in some sort of relationship. In this way the player is considered a controller. As we can see, every object is connected to the player object. If the player object doesn’t exist, then most of the objects doesn’t do anything. The only object that can be considered the exception is the title screen, and logo object. There is a relationship between these two, but it can exist without the player object. We can open the video game, and just open the title screen. It won’t do anything, but we can exit the game easily without actually playing the game.

As I said earlier, almost all the objects in the class diagram cannot exist without the player object. We can see that most of the relationships with the objects are composition. They cannot technically exist without the player actually interacting with them. For example, the pause object will not do anything unless the player presses the escape key. The pause object won’t freeze the game in the title screen, it would only do so when the player object exists. We can go on with the gameover object. It is only activated when the player object falls off the stage. The checkpoint is only changed when the player object collides with it. The room is another special case, the player object spawns there, but the room can exist without the player object. You can make a room, with many traps, but other than that, the room is only used to make sure the player doesn’t fall off screen.

Because of this the class diagram is high cohesion, and high coupling due to the high tight the diagram is. Because of this, the code will be hard to reuse, because a video game is a very specific type of program. A database for alcohol inventory would be more manageable and easier to maintain then a video game. However, because it is a video game, this cannot be helped.

Now while it is true that it is hard to maintain. I will also point out that the player object doesn’t create the objects. They are directly interconnected with many of them called in the player object using global variables and such. We can see from the sequence diagrams, that the player doesn’t technically create the objects. The player object does call on it, but the objects themselves can exist, but cannot be called upon without the player object. It is a very close relationship that cannot exist without the player object. That is why the player object is the controller, as it delegates through everything. With a press of a key, an object is called to affect the system in some way or another. We can see this clearly through the sequence diagrams, and the class diagrams above.

In conclusion, we can assume from the class diagram and the sequence diagrams that the player object is the controller. The other object relies on the player object to delegate when to use the objects in the system. This is done with keyboard or video game controller inputs from the actual player.