**Design Documentation for Heli-Rescue**

**Nick Kowalchyk, Caroline Bonomo, Ryan Gloff, Alex McGrath, Justin Buck**

**Environment**

**Design**: The environment should be capable of holding the player the terrain and flying entities as well as any future object that could be used in the game.Terrain should be at the bottom of the screen, flying enemies at the top, and player in the middle.

**Implementation**: Environment is a class that holds all of the entities that are part of our game. It currently holds the player object, the terrain and the flying entities above the player.

**Constraints**: None

**Testing:** Player can go off screen; **Player can no longer go off screen**

Frame rate drops as game runs; **resized buffer after deleting the objects pushed to screen.**

Tested Constructor - initializes and returns expected values

**Terrain Generation**

**Design:** (originally planned as one, continuous hill) generation of a seeded random set of obstacles (skyscrapers, pterodactyls, etc.) to screen

**Implementation:**

**Testing:**

Tested Constructor - initializes and returns expected values

**Input**

**Design**: Single input class that handles key and mouse input for all aspects of project (player movement, menu handling)

**Implementation**: Class currently handles player movement, through use of position setter functions in object class; currently not implemented with menu screens

**Constraints**: None

**Testing:** When player presses “esc” game closes and continually reopens and then closes, if you press and hold up then press down and release down you still go down

**Menu Screens**

**Design**: Simple screens to greet you in the game and to warn you when you are exiting

**Implementation**: Class currently has a very basic menu screen/exit screen.

**Constraints**: Handles .ttf fonts only

**Testing**: The menus would crash the game if left idle; **change code to close the font file after being opened**.

**Rescue/Crash Collisions (within Object Class)**

**Design**: Collisions should be handled between the player and the terrain the player and the flying entities and the entities and the terrain.

**Implementation**:The player colliding with the terrain causes a print to standard output saying the player has collided with the environment. The player colliding with and entity causes the entity to be destroyed and the a collision between an entity and the terrain results in the flying entity to be destroyed.

**Constraints**: This should be implemented as efficiently as possible to reduce lag on the system.

**Testing:** Output to the console displays when a player has colliding with terrain, an entity, or player immediately when the collision happens and is happening at the correct times.

**Image Loading**

**Design**: Extension of the object class to able to add images to objects

**Implementation**: image files can be added to an object so that the player can have a visual representation of that object

**Constraints**: image type can only be .bmp

**Testing:**

**Sounds**

**Design**: class to load sound files and assign them to different parts of the game

**Implementation**: when the player collides with objects a sound will play to let them know. Background music is also added to give them player music to listen to while playing.

**Constraints**: files can only be .wav

**Testing:**

**Player Failure/Scoring**

**Design:** class to handle displaying the player lives and the player's score. The player is given three lives

**Implementation:** if a player picks up a person then their score will increase by one. If the player collides with an entity or obstacle then their lives will decrease by one till it reaches zero.

**Constraints:** none

**Testing**:when the player loses all three lives they are brought back to the menu screen but are not able to restart the game. Scoring has a max of 1410065408 once reached it does not add anymore.

**For Presentation**

Design Artifacts

Testing shown based on the design

Stuff you can deal with in mass

Mostly automated

General demo