Game Project: Bullet Hell

Student: Wensheng Chen

ID: 0390575

This document specifies the key points and features that I programmed for the project. The following are the important classes and features I am going to discuss:

1. External map
   1. XmlTag – specifies the basic structure of a single markup tag
   2. XmlStruct – specifies a structure and hierarchy of tags for a XML file
   3. XmlValidator – validates the given file according to a XmlStruct
   4. MapXml – defines the XmlStruct for an external map
   5. MapLoader – loads the external map file that contains the sources of audios, graphics, objects, and events into a Map object
   6. Map – contains the audio, graphics, objects, and events that are required for a stage to be executed

The ability to load an external file into a map that is executable by the game is a major plus for the project. It allows any user, with the ability to write a map XML file, to externally customize the map, change the settings, and even create new maps for the game. This is a very important feature if the product were going to the market, where the user base is usually over a few hundred-thousand or even millions. In developer’s perspective, this is extremely convenient as the game allows flexibility without recompiling the project each time there is a change in the map.

1. Behavior model
   1. Behavior – the abstract base class for all other behaviors
   2. BehaviorController – controls the behaviors; what should an enemy plane object do in different circumstances
   3. ShootingBehavior – controls what type of bullet to shoot
   4. Behaviors implemented:
      1. Chase
      2. Evade
      3. Linear movement
      4. Self-Destruct
      5. Shoot Straight – shoots at 270 degree
      6. Shoot Circle – shoots a bullet at every few degree over 360 degrees around the shooting plane object
      7. Shoot Target – shoot at an angle that aims the target
      8. Shoot Straight Angle – shoots straight at a given angle

The behavior model is the second highlight of this project. Behavior is crucial to plane object’s flexibility and artificial intelligence to achieve a specific goal. The behavior model in this project is set up in a way that different planes can share a common behavior. This is achieved simply by having a reference to an instance of that particular behavior class. Therefore, an enemy plane object can have multiple references to different behaviors to grant the object’s AI requirement. With this behavior model in hand, I will be able to create different sets of behaviors for a particular object without recoding the same statements over.

1. Game state manager
   1. GameStateManager – controls the transition of one state to anther
   2. GameState – Abstract base class for a state
   3. Game states implemented:
      1. Dialog – creates dialog boxes that dynamically wraps around the given string
      2. Menu – controls the flow of game state
      3. StartGame
      4. Setting
      5. Opening
      6. Map
      7. Credit
      8. Continue

Game state manager is critical to the process of development for a game. A well built and structured game state manager can ease the job of a developer by factors. The game state manager provides the ability to separate game states or screens into different classes. Separation of classes improves readability, maintainability, and thus, the progress of development for a large project. The game state manager in this project uses classical and object-oriented hierarchy approaches to transition game states.

1. Multiple Abstraction Layers
   1. Obj – defines a basic object
   2. MoveableObj – defines an Obj that has the ability to move
   3. MoveableSpriteObj – defines a MoveableObj and animates the sprite as it move
   4. PlaneObj – defines a MoveableSpriteObj with the ability to project BulletObj and behave
   5. Classes that were implemented based on the above abstractions:
      1. BulletObj – defines a MoveableObj with the ability to hit does damage to the target
      2. EnemyBulletObj – defines a BulletObj with a specific target
      3. HeroBulletObj – defines a BulletObj with a wide range of targets
      4. EnemyPlaneObj – defines a non-playable PlaneObj and is controlled by Behaviors
      5. HeroPlaneObj – defines a playable PlaneObj
   6. Other Abstractions:
      1. Behavior
      2. XmlStruct
      3. GameState

Multiple abstraction layers has the advantage of writing class-specific code, in another word, it reduces the amount of redundant codes the developer has to write. It also has another advantage of enforcing the developer to writing abstract method that is used for polymorphism. The abstraction in this project proofs useful for future add-ons.   
(*Note: for some classes the abstraction was removed to be compatible with the external map*.)

1. Reusability
   1. The abstractions can be reused in other games.
   2. The game state manager can be reused as well.
   3. The concept of behavior model can be reused.
   4. The architecture of external map can be reused.
   5. The classes in System directory can be reused, specifically the following:
      1. Dialog
      2. Event
      3. MenuNode
      4. TimeEventHandler
      5. InputHandler
      6. Menu – can be reused with some change

As taught in software engineering class, reuse of a reliable artifact or code can reduce time and budget in development and maintenance. The project shows a great percentage of reusable codes.

As requested, the following are examples of bullet hell series game and a download link for Touhou Project if closer look is needed. Touhou Project is a series of bullet hell games.

<http://www.youtube.com/watch?v=4aS-D4s6XSg>

<http://www.youtube.com/watch?v=s8JmfCYmtHo>

<http://www.youtube.com/watch?v=bXNojt3NF8g>

Game download link:

<http://www.wuala.com/solracklol/Touhou/Juegos/Oficiales/Juegos%20Touhou%20Windows/>

I previously downloaded the game from another source; I cannot guarantee the reliability of the source.

*Note: 10.5 Scarlet Weather Rapsody is a game of different genre, specifically 2D versus game.*

I have included an online documentation of my game created by doxygen. Open index.html in the folder for reference.