R. Arietta
CIS586 F2014
Assignment 1

Part B - Technical Specifications

For my first assignment, I have chosen to deconstruct the game Frogger. The following technical specifications provide the necessary details concerning my intended implementation of the classic gameplay.

1. **GAME DESIGN:**

   a. **Player goals and objectives**
   The goal of this game is for the player to maneuver each of a series of frog characters vertically across the screen. This involves navigating several busy lanes of traffic and an obstacle-filled river in order to reach the designated safe havens on the opposite side.

   b. **Challenges and conflict**
   In their attempts to reach safety, the frog characters face several challenges and obstacles along the way. When crossing the lanes of traffic, there are many vehicles (cars, trucks, etc.) traversing the screen horizontally. Colliding with any of these vehicles while on the road would result in the frog dying and the user losing a life. The river presents another source of conflict. The frog character cannot move onto a space occupied by water unless there is a floatation device present on that space (i.e. a log or a turtle). These floatation devices present further challenges, as they are not static. The logs and turtles all move horizontally across the screen and will carry a frog character that jumps on top of it, and the turtle agents also dive under the water periodically, resulting in the frog sinking.

   c. **Constraints and boundaries**
   The player is constrained by several gameplay features. The first of these is the screen area of the game. If the frog character travels off the screen, either by user control or while traveling on a floatation device on the river, the frog character will die and the user will lose a life. Another such constraint is the time of gameplay. Each frog has a limited amount of time to arrive safely across the river, which is indicated by a labeled progress bar on each level screen. The final gameplay constraint is the number of lives a user has. When the game starts, the user is allotted a set number of lives (3). If the user's frog characters meet their unfortunate end this number of times, then the game ends and the user is presented with a GAME OVER screen, after which they can choose to restart or quit the game.

   d. **Resources**
   The base gameplay includes limited resources with which the user can interact. The only resource present is the "extra life" or "1-UP" object that randomly appears in the safe havens across the river (not to be confused with the "1-UP" text heading indicating Player 1 gameplay). If a user is able to land a frog character in the safe haven when this object is present, then he/she will be rewarded with an additional life (up to a maximum of five).

e. **Detailed description of the rules, including win/loss conditions.**
When the game is started, the user is presented with a level scene. The level scene will feature a frog character at the base of the screen. Using the arrow keys, the user can move the frog in any of the cardinal directions one unit at a time. The user must cross the scene vertically without being killed by any of the obstacles. The user can die in any of the following ways:

1) Getting hit by a vehicle on the road
2) Falling in the river
3) Running out of time
4) Going off screen (either while on a floatation object or by user control)

If any of these conditions is met, the user will lose a life. The loss of all the allotted lives will result in the end of the game and the loss of the user. If the user is successfully able to navigate the scene vertically, he/she will find a fixed number of safe haven spaces on the opposite side of the river. The user must direct the frog character into one of these spaces, after which the frog will be locked in place and a new frog will appear at the base of the gameplay screen. The user must successfully guide these subsequent frog characters into the safe haven spaces until all such spaces are filled. At this point, the user will have victoriously completed the level, and a new level will be presented for the continuation of gameplay. Points are awarded to the user using the following heuristic:

1) 10 points per successful space moved
2) 50 points for every frog that arrives home safely
3) 1000 points for completing a level (returning all five frogs to their homes)
4) 10 points per remaining second in the timer upon the successful return of a frog to a safe haven

2. **SCENE DESCRIPTIONS:**
The scenes contained in Frogger's classic gameplay can be categorized in three basic scene types. Each of these scenes and their gameplay objects are discussed in more depth below.

a. **Start Screen**
The first of these is the startup screen, which allows some limited interactivity for the user to start the game or quit. There is only one startup screen scene to the game, and the animation will loop until an option is chosen.

i. **Start Screen Game Object List:**
1. 1UpScoreGUI
2. HiScoreGUI
3. CreditsGUI
4. PointTableGUI
5. StartGameGUI
6. FrogObject (7)

ii. **Start Screen Player Input**
1. Clicking the StartGameGUI initiates gameplay and opens Level 1

b. **Level Screens**

The second scene type is the level scene. There can be one or more levels present in the course of gameplay, depending on how many distinct level scenes are designed. These level scenes are where gameplay actually happens, and where the user maneuvers the frog characters as discussed above.

i. **Level Screen Game Object List**

1. 1UpScoreGUI
2. HiScoreGUI
3. TimerGUI
4. LevelGUI
5. FrogObject
6. GlobalObject
7. GlobalVehicleObject
8. GlobalLogObject
9. GlobalTurtleObject
10. Road pieces
11. Grass pieces
12. Water pieces

ii. **Level Screen Player Input**

1. UP key maps to manual Frog script module that moves the Frog character vertically up one space in the game screen
2. DOWN key maps to manual Frog script module that moves the Frog character vertically down one space in the game screen
3. RIGHT key maps to manual Frog script module that moves the Frog character horizontally right one space in the game screen
4. LEFT key maps to manual Frog script module that moves the Frog character horizontally left one space in the game screen

c. **Game Over Screen**

The final scene type is the loss or “game over” screen. Like the startup screen, there is only one version of this scene, and it is encountered when the user dies too many times and the game is lost.

i. **Game Over Screen Object List**

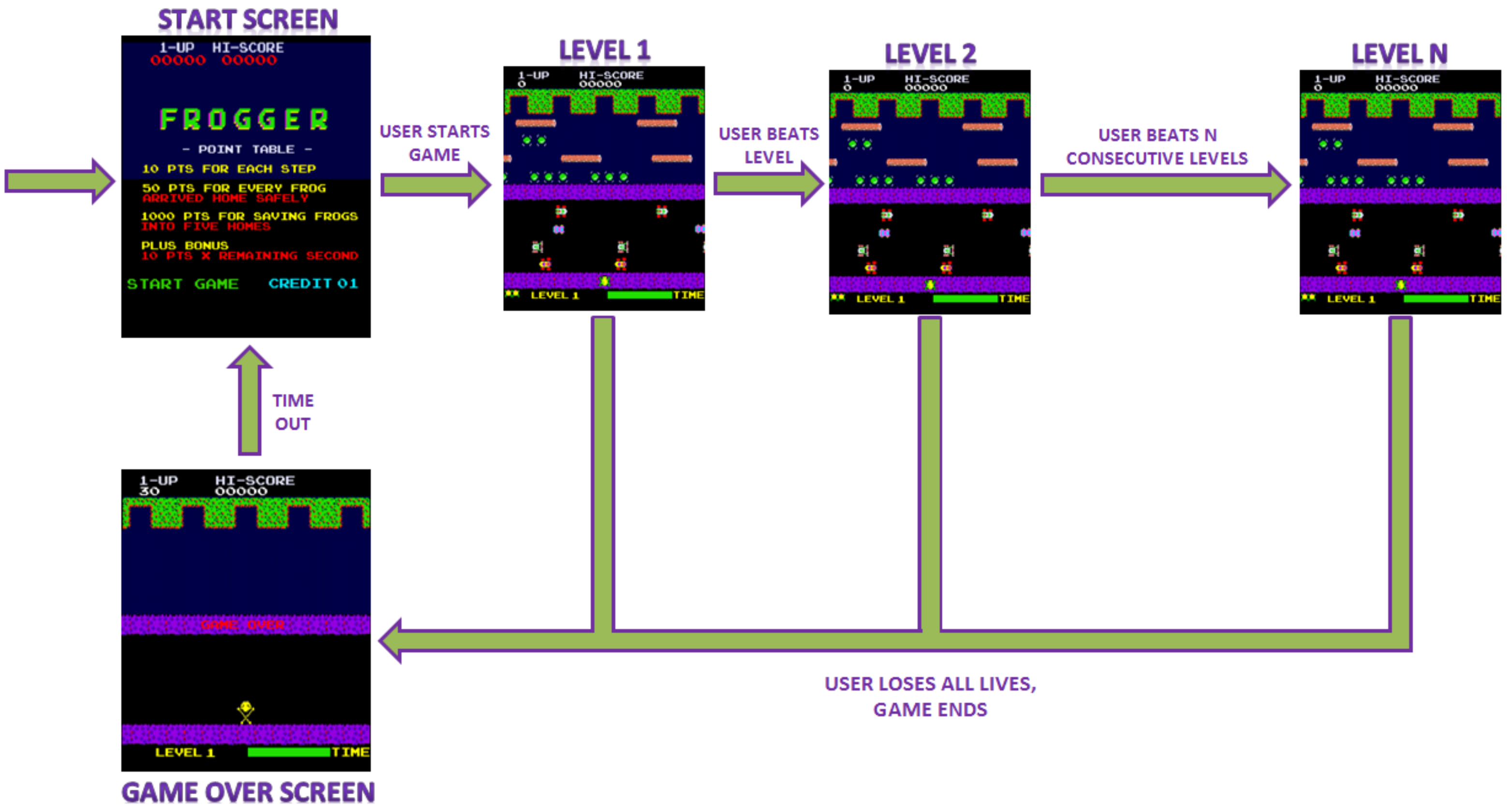
1. GameOverGUI

ii. **Game Over Screen Player Input**

1. N/A, short automatic timeout returns to the start screen

3. SCENE CONNECTIVITY:

The following connectivity graph illustrates how control is transferred from one scene of the game to another.



4. **OBJECT/PREFAB DESCRIPTIONS:**

   a. **FrogObject**

      i. Assets

         1. **frogTexture_Base**: This is the basic texture image of the frog character that is displayed during gameplay.
         2. **frogTexture_Extended**: This is the texture image for the frog character that is displayed briefly when the character is moved in any direction by the user. The switch between this and the base will give the illusion of hopping, just as in the original gameplay.
         3. **movementSound**: This is the sound that is played every time the frog character is moved by the user.
         4. **deathSound**: This is the sound that is played every time the frog character dies during gameplay.
         5. **victorySound**: This is the sound that is played every time the frog character successfully reaches the home or safe haven spot on the game screen.

      ii. Standard Components

         1. Transform: Used to specify the position and orientation of the frog character
         2. Mesh: The base geometry and renderer of the frog character. In my implementation, it will be a mesh cube.
         3. Rigid Body: Used to detect collisions and interactions between the frog character and the various obstacles and environment units.

iii. Custom Components
   1. Frog (Script):
      a. This script will include the following variables:
         1) **jumpLength** – the distance moved by the frog on each user input
         2) **heading** – the current orientation of the frog character
         3) **rotationAngle** – the amount the frog character needs to rotate each time the heading changes
      b. This script will include the following modules:
         1) **Start()** – sets up variables listed above
         2) **FixedUpdate()** – moves the frog character according to the user input
         3) **Update()** – empty for now, all positional updating is done in the FixedUpdate() function in order to correctly deal with the rigid body
         4) **OnCollisionEnter()** – detect collisions with other objects in the scene after each move and examine the type; if the type is deadly for the frog, it dies; if the type is friendly, then the appropriate point value is added to the score
         5) **Die()** – this function kills the frog and restarts the game

b. VehiclePrefab

i. Assets
   1. **vehicleTexture**

ii. Standard Components
   1. Transform
   2. Mesh (cube)
   3. Rigid Body

iii. Custom Components
   1. Vehicle (Script):
      a. This script will include the following modules:
         1) **Start()**
         2) **Update()**– called every timestep to update the position of the log
         3) **Die()** – called upon collision of the frog with a hostile game object

c. LogPrefab

i. Assets
   1. **logTexture**

ii. Standard Components
   1. Transform
   2. Mesh (cube)
   3. Rigid Body

iii. Custom Components

1. Vehicle (Script):
    a. This script will include the following modules:
        1) **Start()**
        2) **Update()** – called every timestep to update the position of the log
        3) **Die()** – called upon collision of the frog with a hostile game object

d. TurtlePrefab

i. Assets
    1. **turtleTexture**

ii. Standard Components
    1. Transform
    2. Mesh (cube)
    3. Rigid Body

iii. Custom Components
    1. Turtle (Script):
        a. This script will include the following modules:
            1) **Start()**
            2) **Update()** – called every timestep to update the position of the turtle
            3) **Die()** – called upon collision of the frog with a hostile game object

e. GlobalObject

i. Assets
    1. N/A

ii. Standard Components
    1. Transform

iii. Custom Components
    1. Global (Script):
        a. This script will include the following variables:
            1) **timer** – time remaining for the current frog
            2) **level** – current level
            3) **score** – current player's score
            4) **hiscore** – overall high score stored by the game
            5) **livesRemaining** – current number of lives left
            6) **frogsSaved** – keeps track of the number of frogs successfully navigated to the home spaces across the river
        b. This script will include the following modules:
            1) **Start()** – initializes all the global variables listed above
            2) **Update()** – called at each time step to update the time dependent global variables listed above
            3) **Win()** – checks if all the frogs have been delivered to the home spaces; if so, the level is cleared and the next level commences

f. **GlobalVehicleObject**
   i. Assets
      1. **VehiclePrefab**
   ii. Standard Components
      1. Transform
   iii. Custom Components
      1. GlobalVehicle (Script):
         a. This script will include the following variables:
            1) **spawnPeriod**
            2) **vehicleSpeed**
            3) **numberSpawnedEachPeriod**
         b. This script will include the following modules:
            1) **Start()** – initializes all the global variables listed above
            2) **Update()** – called at each time step to update the time dependent global variables listed above; if the global **timer** variable reaches a new period, then this module instantiates a new **vehiclePrefab** traveling horizontally along the road at the **vehicleSpeed** stored for the level

g. **GobalLogObject**
   i. Assets
      1. **LogPrefab**
   ii. Standard Components
      1. Transform
   iii. Custom Components
      1. GlobalLog (Script):
         a. This script will include the following variables:
            1) **spawnPeriod**
            2) **logSpeed**
            3) **numberSpawnedEachPeriod**
         b. This script will include the following modules:
            1) **Start()** – initializes all the global variables listed above
            2) **Update()** – called at each time step to update the time dependent global variables listed above; if the global **timer** variable reaches a new period, then this module instantiates a new **logPrefab** traveling horizontally along the road at the **logSpeed** stored for the level

h. **GlobalTurtleObject**
   i. Assets
      1. **TurtlePrefab**
   ii. Standard Components
      1. Transform
   iii. Custom Components
      1. GlobalTurtle (Script):
         a. This script will include the following variables:

        1) **spawnPeriod**
        2) **turtleSpeed**
        3) **numberSpawnedEachPeriod**
    b. This script will include the following modules:
        1) **Start()** – initializes all the global variables listed above
        2) **Update()** – called at each time step to update the time dependent global variables listed above; if the global **timer** variable reaches a new period, then this module instantiates a new **turtlePrefab** traveling horizontally along the road at the **turtleSpeed** stored for the level

i. Water Space
- i. Assets
    1. **waterTexture**
- ii. Standard Components
    1. Transform
    2. Mesh (cube)
    3. RigidBody
- iii. Custom Components
    1. N/A

j. Grass Space
- i. Assets
    1. **grassTexture**
- ii. Standard Components
    1. Transform
    2. Mesh (cube)
    3. RigidBody
- iii. Custom Components
    1. N/A

k. Road Space
- i. Assets
    1. **roadTexture**
- ii. Standard Components
    1. Transform
    2. Mesh (cube)
    3. RigidBody
- iii. Custom Components
    1. N/A

l. Home Space
- i. Assets
    1. **homeTexture**
- ii. Standard Components
    1. Transform
    2. Mesh (cube)

            3. RigidBody
        iii. Custom Components
            1. N/A

m. 1UpScoreGUI

- i. Assets – None
- ii. Standard Components
    1. GUIText
    2. Transform
- iii. Custom Components
    1. 1UpScore (Script): This script will include a Start() module and an Update() module that communicates with the GlobalObject's global **score** variable, much like in the tutorial.

n. HiScoreGUI

- i. Assets – None
- ii. Standard Components
    1. GUIText
    2. Transform
- iii. Custom Components
    1. HiScore (Script): This script will include a Start() module and an Update() module that communicates with the GlobalObject's global **score** variable, much like in the tutorial. Unlike the tutorial Update() module, it will only update the displayed score if it is higher than the GlobalObject's global **hiscore** variable.

o. TimerGUI

- i. Assets – None
- ii. Standard Components
    1. GUIText
    2. Transform
- iii. Custom Components
    1. Timer (Script): This script will communicate with the GlobalObject's **timer** variable to display the time remaining for the current frog character

p. LevelGUI

- i. Assets – None
- ii. Standard Components
    1. GUIText
    2. Transform
- iii. Custom Components
    1. Level (Script): This script will communicate with the GlobalObject's **level** variable to display the current level

q. LivesGUI

- i. Assets – None

    ii. Standard Components
        1. GUIText
        2. Transform
    iii. Custom Components
        1. Lives (Script): This script will communicate with the GlobalObject's **livesRemaining** variable to display the current level

r. **Extra Life**
    i. Assets
    ii. Standard Components
    iii. Custom Components

5. **INTER-OBJECT COMMUNICATION:**
    a. The **FrogObject** must communicate with the **GlobalScoreObject** in order to update the global **score** variable whenever a move is made
    b. The **FrogObject** must communicate with all geometry assets in the scene in order to determine the type of geometry during an OnCollisionEnter() event and respond appropriately
    c. The **TimerGUI** object must communicate with the **GlobalScoreObject** to appropriately display the time to the user via the **timer** variable
    d. The **1UpScoreGUI** object must communicate with the **GlobalScoreObject** to appropriately display the current score to the user via the **score** variable