# Game Design Document Template

1. Title Page
   1. Game – Frogger – The Challenges of Frog Life  
      Frogger – A frog attempts to cross cars passing on a highway as well as through a river to reach safety
2. Game Overview
   1. Game Concept
      * A player controls a frog and attempts to cross several hazards presented by passing cars and crossing a river by making use of logs floating in the stream.
   2. Genre
      * Platformer
   3. Target Audience –
      * Gamers of all ages that enjoy casual gaming experiences
      * Age Group: ESRB rating – E
      * Types of Gamers: Casual Gamers and beyond who enjoy casual and non-intense platformers.
   4. Game Flow Summary – How does the player move through the game?
      * The player can move in 4 directions with the UP, DOWN, LEFT and RIGHT arrow keys. Avoiding hazards as they go.
   5. Look and Feel – What is the basic look and feel of the game?
      * This game is a replica of the original Frogger game which was one of the earliest platformers. As such the game aims to replicate this feel with low res 2D textures in a 2D top down game space.
3. Gameplay and Mechanics
   1. Gameplay
      * Game Progression
        + The player can move in 4 directions with the UP, DOWN, LEFT and RIGHT arrow keys. They spawn on one side of the map and need to cross several hazards. The first is a highway with three lanes of cars. The player needs to take advantage of the gaps between the cars to safely cross. They are then presented with the final hazards which are logs travelling in a river with the stream. The player needs to jump between the logs to safely clear the river and complete the game.

* + - Mission/challenge Structure
      * Challenge comes from the random nature of speed and direction of travel of the logs and cars. Since the distance between objects and their speed vary it creates a varying challenge for the player to be able to circumvent them all.
    - Puzzle Structure
      * The player is not presented with any puzzles as they would normally be viewed since the speed and distance of objects is designed to be random in nature.
    - Objectives – What are the objectives of the game?
      * The objective in frogger is to get to the other side of the map without dying.
    - Play Flow – How does the game flow for the game player
      * The play has a top down perspective and makes continuous progress forward until they die at which point they are reset to the start.
  1. Mechanics – What are the rules to the game, both implicit and explicit? This is the model of the universe that the game works under. Think of it as a simulation of a world, how do all the pieces interact? This actually can be a very large section.  
     + Physics – How does the physical universe work?
       - The expression of physics in this game is the logs in the flowing stream. The logs move across the stream as they are pushed by the flow of the river. When the player jumps on to the logs they too are pushed along the river.
     + Movement in the game
       - The game has a top down perspective and the player Is allowed to move in the 4 cardinal directions which the player can do with the UP, DOWN, LEFT and RIGHT arrow keys.
     + Economy – What is the economy of the game? How does it work? (Lives and score)
       - The player earns score by making forward progress and their total progress is tracked. This is done to prevent them from being able to move forward, back and then forward again to gain extra points. Instead they need to make it further than they have before to earn more points. They also get a big bonus when reaching the other end of the game.
       - In the games current state the player has unlimited lives however they should lose points when they die.
     + Screen Flow -- A graphical description of how each screen is related to every other and a description of the purpose of each screen.
       - The current design of the game does not overcomplicate the style of game with way to many screens. The executable launches into the main game loop which is the only screen presented to the player outside the desktop where they use the executable. While outside of standard convention this allows someone to start playing immediately and as it is a casual game that someone is unlikely to play for a long period of time it means that there are far fewer quit moments for the player to encounter.

Desktop

Main Screen (Game Start/Title)

Screen Flow Diagram

* 1. Game Options – What are the options and how do they affect game play and mechanics?
     + The player has access to no options

1. Levels
   1. Levels. Each level should include a synopsis, the required introductory material (and how it is provided), the objectives, and the details of what happens in the level. Depending on the game, this may include the physical description of the map, the critical path that the player needs to take, and what encounters are important or incidental.
      * There is only a single level. There is no single critical path as the player has to find a route to the end based on varying hazards that are randomized to a degree.
   2. Training Level?
      * There is no tutorial.
2. Interface
   1. Visual System. If you have a HUD, what is on it? What menus are you displaying? What is the camera model?
      * The HUD should only display the score to the player leaving as much screen space clear as possible.
   2. Control System – How does the game player control the game. What are the specific commands?

Joystick inputs to move player (you can also add keyboard options)

* + - * There are 4 inputs the player has access to
        + UP
        + DOWN
        + LEFT
        + RIGHT
  1. Audio, music, sound effects
     + There are 4 music tracks that loop through in a fixed order
     + There are sound effects for when the player is squished by a car as well as when they fall into the river

1. Artificial Intelligence
   1. Opponent and Enemy AI –
      * There is no AI in the shape of a state machine, neural net or even a logic tree. The AI is simply composed of several hazards which move at varying rates to create randomization. They have no logic beyond this function.
   2. Support AI -- Player and Collision Detection
      * There is collision detection between the player and the cars to know if the player has been squished by a car.
      * There will be collision detection between the player and the river to know if the player has fallen in
        + There is another check to ensure that the player is NOT on a log when they collide with the river, if they have the player will not die as they are currently on a log and thus safe.
2. Technical
   1. Challenges
      * The primary challenge in this project is presented by the 2D environment as well as the use of SDL as opposed to a modern engine or modern tools. SDL exposes less variables to the player and has less feature sets built in. As a result a lot more work must be done to create classes that have the functionality that is standard in the industry. However, this also presents an opportunity for learning and applying C++ OOP concepts as everything from helper functions to collision detection and game objects must be built from the ground up instead of being present in the system beforehand.
   2. Classes used
      * CAPP
        + This implementation was designed by Yash and provided for the project ahead of time. It server as the main class of the game from which all basic functionality such as Init, Loop, Render and Events are executed.
        + AssetManager
          - I built this class as a helper class to make adding assets easy by creating functions that spit out completed file paths. The big benefit here is time saving as well as a huge amount of time saved as I changed the file directories later and only had to change the relative file path and all other assets were updated accordingly.
        + Car / Log
          - The Car/Log object are both inheriting from Sprite to allow easier spriteloading however, they do have a lot of recycled code. In a perfect world there would have been a class higher in the higherarchy that inherited from Sprite and was then inherited by Car and Log since so much of their functionality is identical.
          - With their current design Car / Log are the two primary objects to be instantiated. They include nearly all of the functionality for the game objects the player interacts with.
        + MusicPlayer
          - This class handles all of the Music and Sound Effects that are present in the game.
        + Object Manager
          - The Object Managers purpose is to handle rendering, collision detection and any logic that requires knowledge of all the instantiated game objects. If it’s in the game and requires any input the Object Manager handles it.
        + Player
          - The player class holds all logic and variables that relate to the Player object.
        + Score
          - The Score class holds all logic that relate to the score system.
        + Sprite
          - The sprite class manages the functionality of game sprites.
   3. Concepts used
      * OOP Concepts
        + Modular – Component based system
          - Functionality is divided into their own classes that centralize related logic and functionality. This approach makes it easy to add or remove components as well as integrate them as they are designed to be as self contained as possible.
        + Inheritance
          - Several objects are inheriting from others displaying this concept. As an example both Car and Log inherit from Sprite.
        + Virtual / Override Keyword
          - Virtual functions and override functions are used and displayed in the project
        + Object Oriented Approach
          - The modules are built such that the systems and managers need only be instantiated once and their functionality used but are self contained to them. This allows them to be instantiated by the main class CAPP once and function correctly.
          - The game objects are also their own classes designed to be instantiated as many times as necessary with all relevant logic stored within such that they function as required.
3. Game Art – Key assets, how they are being developed. Or Free assets? Free assets found on internet can be sited here.
   1. All assets used were free assets sourced online.
      * Music - <https://opengameart.org/content/42-monster-rpg-2-music-tracks>
      * Sound Effects - <https://opengameart.org/content/8-wet-squish-slurp-impacts>
      * Sprites - <https://www.pinclipart.com/picdir/middle/66-663246_frogger-arcade-graphic-frogger-sprite-sheet-clipart.png>
      * Fonts - <https://www.dafont.com/upheaval.font>