Life among Lions

Technical Plan
Ben Strong

Delivery Platform

For this game we are targeting mobile and PC platforms. Under PC, we are targeting primarily Windows users. This is the largest community of gamers on one platform and would be a great platform for an initial release. Because of this we plan to release the game initially on the Steam platform to capitalize on this large market. Our target market is children, and mobile platforms such as phones and tablets have become the most common gaming platform for children, representing 63 percent and desktops and laptops represent 45 percent of the market. Console ports could be considered at a later point in development.

Development Environment

For developing this game, we are going to be using Unity. There is no real need for high quality art, as our goal with the game is to push across the traditional African art. Because of this, a game engine that is capable of high quality light rendering, texture mapping, etc, is not necessary and Unity fits this description. Our programmers are very familiar with the Unity game engine and feel confident producing games from it. Unity allows for creating builds to multiple different platforms such as Xbox, Playstation, Windows, MacOS, and mobile devices, so if the decision is made to port the game to another platform, it could be done so easily.

Game Mechanics and Systems

- Spawning
  - Spawning will work randomly. Two arrays, one with Game Objects and one with spawning locations will be created by the spawner at runtime.
  - Using Unity's Random.Range() static function as our RNG, one instance from both arrays will be used to spawn a random unit at a random
- Redirecting Objects
  - All Objects will randomly generate a velocity at spawn time

    - This velocity can be manipulated by clicking on the object that you wish to manipulate. This will cause the object to pause momentarily. The object will resume moving at a new velocity when the mouse button is released
    - The new velocity is in the direction of where the cursor was when the mouse button was released
- <u>Colliding Objects</u>
    - The goal of the game is keep lions and people from colliding with each other
    - If a lion comes into contact with a herder or poacher, a “life” is lost
    - All collision is done through Unity trigger colliders
- <u>Deleting Objects</u>
    - There will be a destroyer object that has a trigger collider that will delete all objects that leave a certain area
    - This is to ensure that there are never to many game objects in the world at one time
- <u>Win/Lose State</u>
    - We are planning mission based objectives based on time
    - Each mission will have a different spawn rate for each object type
    - Each mission will have a timer that the player must keep at least one life at the end of the timer in order to complete the mission
    - If a player loses 3 lives within the mission’s time limit, the player loses the game
- <u>Aggro System</u>
    - This is the system that determines what happens when one type of object approaches another type of object.
    - Lions will attack herders under certain conditions
    - Poachers will attack Lions

<u>Technical Risk Assessment</u>

The amount of code to show the basis of gameplay is not to much. At most, the basis gameplay, which includes spawning and changing directions of objects, should be no more than two scripts of around 100 lines of code each. The technical difficulty comes with the aggro system and how in-depth we really want to go with it. The system could be as simple as a trigger surrounding the lion and when things enter that trigger, the corresponding events occur. Or the system could be a little more complex, having an aggro variable on both lions and poachers. If one number was high enough compared to the other, the poacher would attack the lion.

Design Pipeline

Workflow will proceed as follows:

- File Modification
    - Designers can set up an SVN repository in order to access the game. From this SVN folder anyone can edit the files needed to add mechanics, etc.
- Version Control
    - All game files will be pushed to the repository on pineapple in the same format that Unity reads them and builds a project. This makes downloading the game for development purposes the same for all working on the project.
- Project Building and Storage
    - The most recent version of all art, scripts, documentation, etc., will be uploaded to the pineapple repository.
    - The project will be built using Unity and uploaded to the pineapple repository.
- Changes Made In Editor
    - All units and spawners have public variables that can be modified in the Unity editor. These variables include unit movement speed, unit spawn rate, . These are in place for the designers to play around with the balance of the game in a way that is quick and easy.

Art Pipeline

Workflow will proceed as follows:

- File Preparation
    - 2D art will be created in Photoshop and saved in .png format. Animations for the characters will be created in Unity, and therefore a sprite sheet will be created for the needed frames of the animation. All 2D art should be the same pixel size of 100 x 100 pixels to one “tile size”.
    - 3D art (as of now has not been discussed for the digital version of the game) will be created in Maya/Blender and saved in .fbx format.
- Version Control
    - All art will be uploaded to the Unity Project through the Assets/Sprites folder in the repository on pineapple. There is only one artist on the team so multiple people editing the same file at once and uploading different versions will not be a problem.
- Project Building and Storage
    - The most recent version of all art, scripts, documentation, etc., will be uploaded to the pineapple repository.
    - The project will be built using Unity and uploaded to the pineapple repository.