Sherpa Life

Technical Plan

EGD-240-83 Game Production I
Team 3 Project 1
1/30/2017 Vedant Chaudhari

1. DELIVERY PLATFORM

Sherpa Life is being developed for mobile platforms such as phones and tablets.

- We are targeting casual gamers
- iPhone owners spend on average more money than other mobile platform owners
- The App Store has the highest median application revenue, with Android in second, and Windows Phone last

With our target audience, iOS and Android are the two best platforms to develop for.

The app will be primarily developed with the Unity Editor on a Mac Laptop. The core mechanics and UI will be designed around touch based interaction. Release candidates can be compiled and built for both iOS and Android directly from the laptop.

2. DEVELOPMENT ENVIRONMENT

Our team will use the Unity engine to maximize productivity and realize the potential of our game prototype. The programmers will be using c# to build the scripting assets. Unity allows us to use a single codebase but deploy to multiple platforms including Android and iOS, we could even deploy to windows phone in the future. Unity also makes it easy to implement social features and in app purchases. c# is a language our developers are very familiar with. It will allow them to quickly develop features, debug, and build our game as fast as possible.

Unity Engine will also make it much easier to develop our game and achieve our vision. General ease of use and the design paradigm of Unity such as prefabs and scripts will help us rapidly prototype gameplay features and mechanics. Bugs can be reproduced and found easily. Unity will enable us to rapidly iterate upon the game concept based on immediate feedback and QA feedback.

3. GAME MECHANICS AND SYSTEMS

This game is a turn based 2D puzzle game. Sherpa Life's core mechanic is to traverse a series of levels fulfilling objectives your tourists create for you. The win condition is achieved when a player fulfills all the objectives and reaches the peak of the mountain first.

The movement mechanic will be RNG based using a dice to determine the number of moves a player has. Core mechanics are simple and robust, they will be easy for the player to pick up and be popular with the casual audience. Additionally, the game has strategy at higher levels, that experienced players will enjoy.

Each level will have four objectives that the player must complete. More objectives per board the lower the player is on the mountain. Certain objectives such as bushes or trees have a single tile line of sight, where the player has to only be within one tile to view the objective. The flower is the only objective which requires the player to stand on it to validate if it is the correct objective.

The players moveset is in eight directions, meaning the player also has the option to move diagonally. The player's amount of moves will be decided at the beginning of each turn through a dice roll that yields a number between 1 - 4. The movement cost of standard tiles is 1, so each turn can traverse 1 of these standard terrain types. Mud is a harsh terrain type that costs 2 turns to traverse 1 tile, hence if the player has 4 turns, he/she will only be able to traverse 2 mud blocks max. The third terrain type is rock, which is impassable.

Certain game changing events can occur such as avalanches that knock all the players back off higher boards to lower ones and make certain terrain impassible for 2 - 3 turns forcing players to either prioritize other objectives or find a way around it.

4. ART PIPELINE

We will design our graphics with dedicated resolution assets since we are primarily targeting the iOS platform. We will use lossless image compression to keep the app size small and under the app store limit for cellular downloads.

Artists will create the art with a 2d Isometric perspective using the same angle as the camera to create the correct perspective. Art will be vectored base so it can scale easily and losslessly for any display resolution.

Our art will be stored within a folder called sprites in unity. The naming convention will be sprite_(name) to make it easy for programmers to identify the art type and for which game objects to add the art to. Artists can add art to this folder by adding, committing, and pushing their assets to the git repository on the artist branch. Thus, the programmers can merge their branch with the master branch when they are ready to implement the art assets.

The board is built using 3d Cubes that have a material assigned to them, with a legacy diffuse shader. Once the artists have developed the art palette for the visuals, the hex values will be used to create materials for each terrain type such as rock, mud, snow, grass, etc. Then these materials will be added to a 3d cube gameobject in order to create prefabs to plug into the tileMap data set.

Sprites for objectives will be attached to quads using a default sprite shader to make the rest transparent and have it display correctly in the game.

5. DESIGN PIPELINE

We will design individual levels manually in scenes in the Unity editor. Obstacles and art assets will be imported and placed manually on a pre generated 2d grid. Designers can make gameplay tweaks and Level tweaks using editor tools and public variable modifiers.

Level design is based on an array called tiles that is declared and instantiated in the main class of the game. The designers can set each array element equal to a number, to represent a specific terrain type from the prefab list in order to build the level.

Designers add objectives to the game by dragging prefabs into the scene and setting the tile coordinates manually from the inspector. Objectives also must be assigned a faction in order to determine if it is intended for player1 or player2.

Each scene will contain anywhere from 2 - 3 levels representing the player climbing up a mountain eventually reaching the peak. As the designer creates multiple scenes, they can all be added to the scenes folder and backed up to the repository from which the programmers can access it.

6. MILESTONE UPDATES

Milestone 1

Our team approached this milestone very well. We built a physical prototype very quickly and got it QA tested to confirm our gameplay mechanics. Our primary development goal is to now implement core gameplay in a digital prototype and have a playable version by next week.

Milestone 2

Digital prototype has a basic movement system and pathfinding down. There still are currently no objectives or a 2nd player, but the player has a fixed movement system that follows a path determined by an algorithm.

Milestone 3

The game is truly multiplayer with 2 players added to the level. Objectives have also been added to the level, as well as an objective UI and score system. Turns have also been implemented. At this stage the prototype has reached feature parity and is a rough realization of our vision.