Feasibility Study

The Group:

Jennifer Green, [Jennifer\_Green@student.uml.edu](mailto:Jennifer_Green@student.uml.edu)

Hansel De La Cruz, [Hansel\_DeLaCruz@student.uml.edu](mailto:Hansel_DeLaCruz@student.uml.edu)

Kyle Jolicoeur, [Kyle\_Jolicoeur@student.uml.edu](mailto:Kyle_Jolicoeur@student.uml.edu)

GitHub Repository: https://github.com/hanselrd/bubble-warrior-adventures

The Client:

RPG fans

The Task to be Undertaken:

The project is to develop a 2D, top-down RPG similar to older games such as Final Fantasy VI and Legend of Zelda. The player will control a single character who journeys back to their home kingdom in an attempt to correct a mistake they made ten years ago. Along the way, they will gain access to stronger weapons and armor, as well as special magical orbs known that bubbles that can confer great power on their wielders.

Designing the game will consist of four main parts: creating and implementing a variety of maps for the player to explore, integrating the story in to the world, and the creation of both a battle and inventory system.

Benefits:

Everyone likes a good 2D RPG, and by using open-source assets we leave us the possibility of selling the game at some point in the future. This program is just for entertainment of the user.

Preliminary Requirements Analysis:

1. Main display
   1. Background
      1. Title / option backgrounds
      2. Map background
         1. Load in maps from files
         2. Load collision (not walkable) areas from map file
         3. Player view
   2. Sprites
      1. Load sprites / textures from files
      2. Player
         1. Animate player movement
         2. Animate player attack
         3. Collision detection
      3. Friendly NPC’s
         1. Text / Lore only NPC’s
         2. Story progressing NPC’s
         3. Currency shop NPC’s
         4. Collision detection
      4. Hostile NPC’s
         1. Animate NPC’s
         2. Casual enemy NPC’s
         3. Boss enemy NPC’s
         4. Enemy projectiles?
         5. Collision detection
2. User Interface
   1. Display interface
      1. Title screen
         1. New game
         2. Load from save
         3. Options
      2. Options screen
         1. Audio options
         2. Resolution options
         3. Full screen/ windowed
      3. Inventory
         1. Equip-able
         2. Slots
         3. Trash items
      4. Main gameplay overlay
         1. Resources (health/mana?)
         2. Currency
         3. Player name
         4. Player level / experience
         5. Player class
         6. Inventory button
   2. Controls
      1. Player movement
      2. Player attack
      3. Menu shortcuts
      4. Saving and loading the game
3. Collaboration
   1. Github
      1. Code sharing
      2. Version control
      3. Code backups
   2. Gitter
      1. Idea sharing
      2. Code issue resolution
      3. Planning for deadlines

Technical Requirements – Feasibility:

1. Images – As of now, all images, tile-sets, and audio will be used from OpenGameArt (website) which can be used royalty-free.
2. Database – Python will store scripts, AI. Possibly will use JSON for saving and loading the game. Tmx file type will store map data including collision areas.
3. Libraries – SFML will handle events, and window display. TGUI will be used for overlaying the main gameplay window, displaying inventory, and for title screen / options windows. Python will store scripts, NPC AI,
   1. Data storing
      1. Python for scripts and NPC AI
      2. Tmx format for map / tile files
   2. multimedia
      1. OpenGameArt website for images, tile-sets, and audio
      2. Tiled map maker / map generator

Scope:

Our goal for this program is to make a fun story and action driven game. We will have currency and shops, battle, player stats, equip-able items and stat changing items, a level up system, upgrading items, enemy NPC’s.

Suggested Deliverables:

Management:

1. *Plot and Script* – a document consisting of the script to the game, listing all dialogue lines and options. Any changes made to the game plot should also be reflected in the script and agreed upon before being put into action.
2. *Design Document* – a document about the design of the system. Document should show the intended flow of gameplay and how the different parts of the game will interact with each other.
3. *Source Code* – a document along with the source code for the game. With this document, the game would be considered complete and ready to play.

Technical:

1. A database that can support our lists of enemies, NPCs, and inventory. Database should be easily expandable to allow for modifications by players.
2. A fully-functioning GUI.
3. A world map with interconnected zones for the player to explore.

Walk-Through: Ten years ago, a young adventurer succeeded in bringing together the four Great Bubbles and defeating an evil that threatened the kingdom. Now, after a successful career as an adventurer, s/he makes the decision to return home and bask in the rewards and adulation that is surely coming to them. However, not everything is right in the kingdom you left behind. Strange beasts have been spotted roaming the land, and entire villages have vanished, never to be seen again. It’s up to you to put right what went wrong ten years ago, and restore peace and harmony to the land once more.

Software Development Process: Our group is using an iterative and incremental approach to game design. Each team member is responsible for an aspect of the game, and at each step the aspects of the game are combined and tested before moving on to the next. If problems are encountered, the team can easily return or redesign in order to solve them.

Outline Plan:

Week 1: Set up Github, dependencies, a basic window, and basic event handling. Download all open domain images/sprites needed.

Week 2 & 3: Moveable character and window’s view of the “world”. Game’s story planned and created.

Week 4 & 5: Character attack, inventory, and items.

Week 6 & 7: Implement structures, and start Enemy AI.

Week 8 & 9: Friendly NPC AI and implementing story/structures.

Week 10 & 11: Find methods to improve efficiency and reliability of the program

Week 12 & 13: Play-tester feedback and implement changes.

Week 14+: Extra time for improvements, missed “deadlines” and implementing new features

Visibility Plan: The group will be meeting twice a week before class in order to catch up in-person on that week’s tasks.

Conclusion: Based on this analysis, the creation of a 2D top-down RPG is feasible, and the group is willing to put in the effort to see our game succeed.