Software Requirements and Design Document

For

Group 10

Version 3.0

Authors:

Reece Gabbett

Ryan Beck

Parker Stone

Timur Bickbau

Marcos Sivira

1. Overview (5 points)

Development of the system is complete with a full overworld, multiple dungeons, a map editor, and a full combat system. The game now has a playable amount of content and can be considered a ‘game’. The overworld has diverse scenery with forests, caves, mountains, beaches, and a giant snake building. Two dungeons, the first one utilized a puzzle that tests lank’s entire weapon arsenal. The second one is a labyrinth similar to the dungeons from the first zelda game. The full combat system includes enemies, sword mechanics, bow mechanics, health, and game overs.

2. Functional Requirements (10 points)

Completed Requirements for Iteration 3:

1. Implemented the full overworld and dungeons. This was a high priority requirement.
2. Implemented player warping to move the player between different parts of the world. This was a high priority requirement.
3. Added arrows that Lank can shoot when pressing X. This was a medium priority requirement.
4. Added signs that display text when hit with a sword or arrow. This was a lower priority requirement.
5. Added switches that are triggered when hit with a sword or arrow. This was a medium priority requirement.
6. Updated the health system to use 3 hearts (including half-hearts) instead of 6 full hearts (with no half-hearts). This was a lower priority requirement.
7. Updated enemies to drop a heart upon death that the player can pick up to heal for one heart (you may need to be a bit further away to see the heart; otherwise you will pick it up automatically). Pots also drop a heart but were not fully implemented. This was a medium priority requirement.
8. Added a pause menu when pressing Esc. This was a medium priority requirement.
9. Added a minimap. This was a medium priority requirement.
10. Added a game over screen when Lank dies. This was a medium priority requirement.
11. Implemented a map editor. This was a high priority.

3. Non-functional Requirements (10 points)

Scalability: Keep FPS locked to 60FPS for stability.

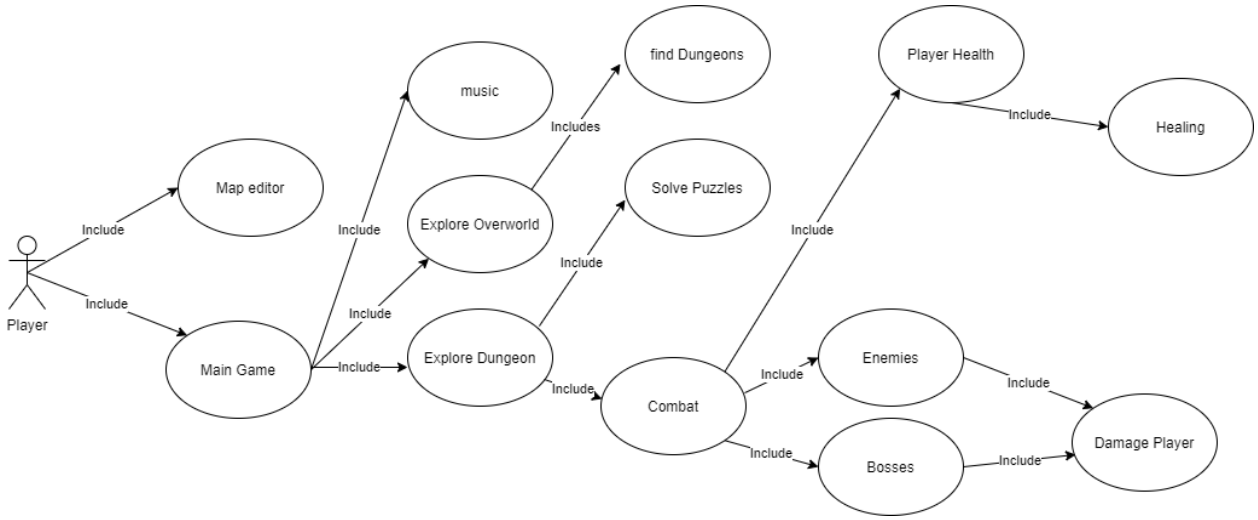
Usability: Run on Windows 7 or later, with keyboard and mouse.

Reusability: Reuse assets (e.g., sprites and movements) for future installments.

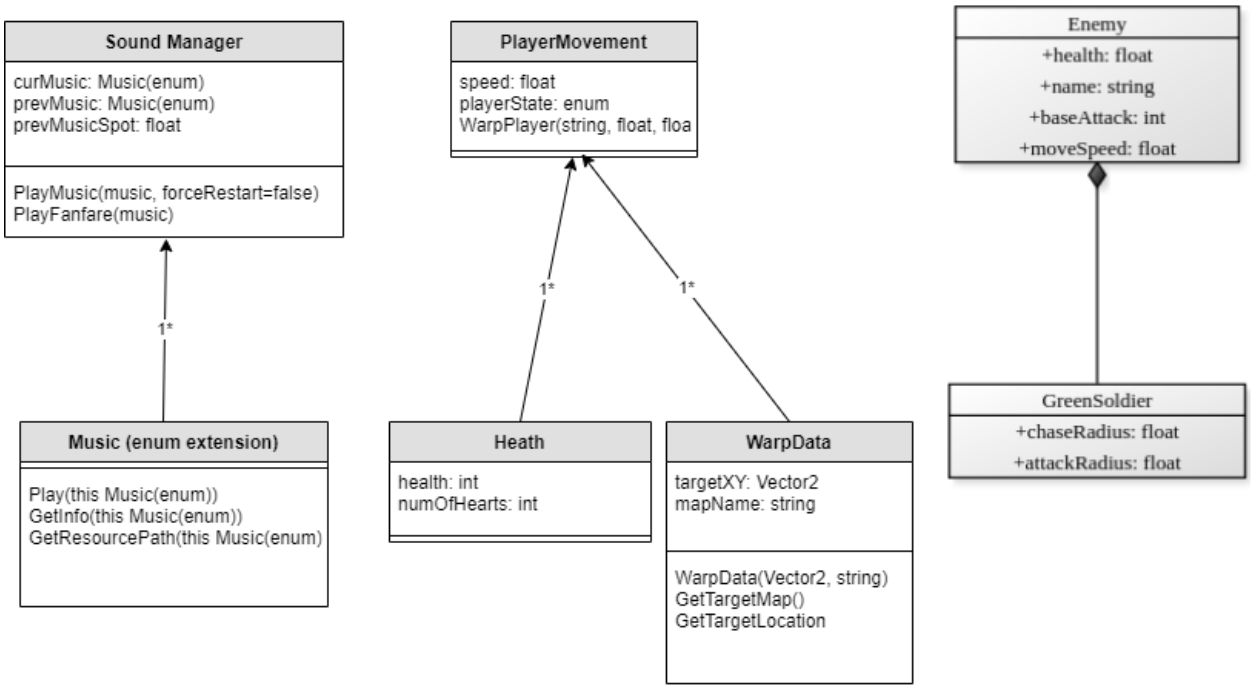
Reliability: Continue to ensure there are no game-breaking bugs.

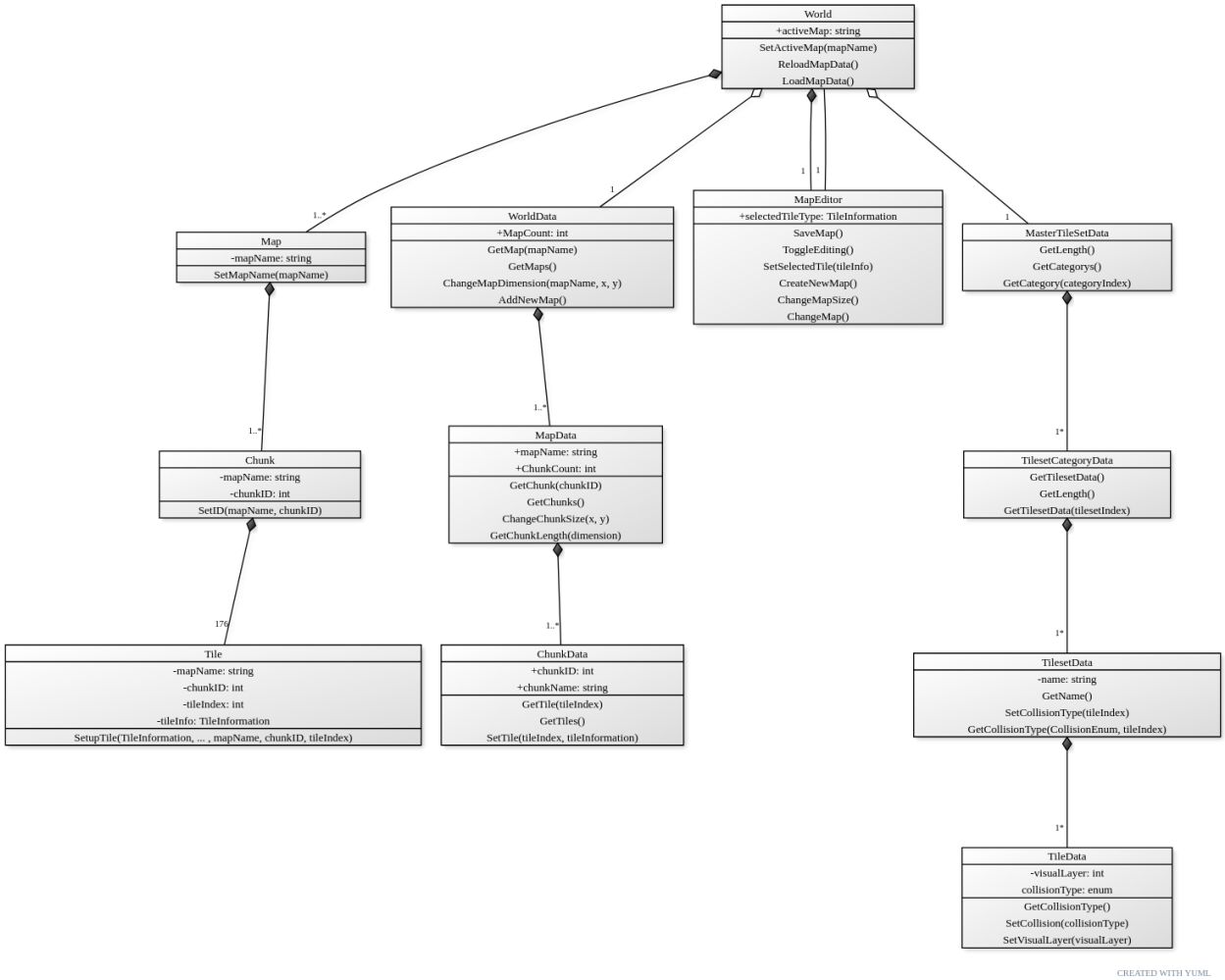
Quality: Maintain a high quality standard for the full game.

4. Use Case Diagram (10 points)



5. Class Diagram and/or Sequence Diagrams (15 points)


Sound Manager
curMusic: Music(enum)
prevMusic: Music(enum)
prevMusicSpot: float
PlayMusic(music, forceRestart=false)
PlayFanfare(music)
1*
Music (enum extension)
Play(this Music(enum))
GetInfo(this Music(enum))
GetResourcePath(this Music(enum)
PlayerMovement
speed: float
playerState: enum
WarpPlayer(string, float, floa
1*
1*
Heath
health: int
numOfHearts: int
WarpData
targetXY: Vector2
mapName: string
WarpData(Vector2, string)
GetTargetMap()
GetTargetLocation
Enemy
+health: float
+name: string
+baseAttack: int
+moveSpeed: float
GreenSoldier
+chaseRadius: float
+attackRadius: float



Notes
Class entries here are simplified.
Map, Chunk, and Tile are MonoBehaviors
WorldData, MapData, and ChunkData are the datas for the respective MonoBehaviors. Used for IO.
MasterTileSetData,..., and TileData are for IO defining general tile info
TileInformation is a specific instance of a tile
Health is for showing in UI
Classes not listed:
TileButton: Gives functionality to MapEditor tile selection
ClickThroughPreventer: """Self-Documenting"""
Knockback: Monobehavior for Giving Knockback.
SongInformation: Stores path, name, and loop point for a song
TileInformation: Stores category, tileset index, and tileIndex for a specific tile instance
MainMenu: UI that either loads the main game scene or exists the application

6. Operating Environment (5 points)

The operating environment is a computer running Windows, using a keyboard and mouse control scheme. The Windows version must be compatible with Unity (i.e., it must be no older than Windows 7).

7. Assumptions and Dependencies (5 points)

It is assumed that the user has a keyboard and mouse, and will be running a version of Windows that is compatible with Unity (i.e., one that is no older than Windows 7).

The game will be dependent upon Cinemachine for cameras, and upon outsourced spritesheets for art design.