Software Project Management Plan

Graze Save Survive

Version 1.0

16 January 2024

https://github.com/holiday-pettijohn/GrazeSaveSurvive (currently private)

Department of Math and Computer Science, Biola University

Overview. This document exists in order to provide a formal definition of the software approach and associated milestones. It also serves to document the agreed deliverables and dates.

Target Audience. Project team members, stakeholders, project manager (the professor assigning the project)

Project Team Members. Holiday Pettijohn, Jacob Shirota, Caleb Zuniga, Nicklas Kuo

Version Control History

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Jacob Shirota	Definition of project overview, deliverables, software project model	01-15-2024
0.2	Holiday Pettijohn, Jacob Shirota	Roles and responsibilities assignment	01-18-2024
0.3	Caleb Zuniga	Tools and Techniques	01-18-2024
0.4	Jacob Shirota	Responsibility Assignment Matrix	01-18-2024
0.5	Holiday Pettijohn	Tasks, Definitions and Abbreviations	01-18-2024
0.6	Jacob Shirota	Assignments (Merged with Tasks)	01-18-2024
1.0		First deliverable submission	01-19-2024

Signatures of Approval

Version 1.0		
Name	Signature	Date
Holiday Pettijohn	Holiday Pettijohn	01/19/2024
Jacob Shirota	Jacob Shirota	01-19-2024
Caleb Zuniga		
Nicklas Kuo	Nicklas Kuo	01-19-2024

Table of Contents

1. Introduction

1.1. Project Overview

Graze Save Survive is a 2D rogue-lite survivor game. It will contain elements of the player entering a game, referred to as a ‘run’, and then trying to make as much progress as they can until they either succeed or fail based on the action combat mechanics. After a run, the player will be able to access upgrades and progressions which will be saved using a database for persistent storage. The player’s objective is to ‘win’ a run with the assistance of the progression upgrades they obtain.

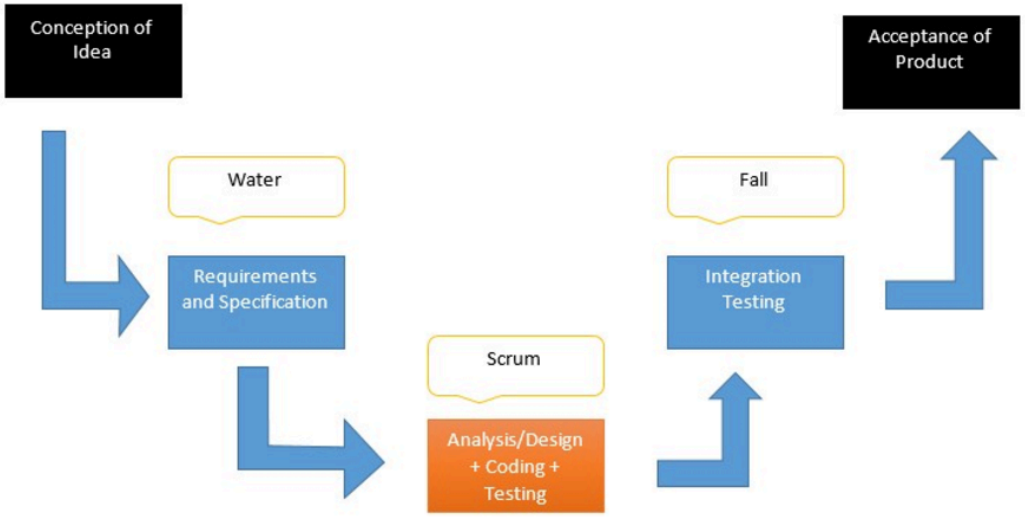
1.2. Project Deliverables

Software Requirements Specification	01-26-2024
Software Design Description	02-02-2024
Scrum Prioritized Product Backlog	02-09-2024
Sprint Artifacts (Sprint Backlog & Burndown Chart)	02-23-2024
	03-15-2024
	04-05-2024
Software Test Documentation	04-12-2024
Project Demonstration	04-23-2024

2. Project Organization

2.1. Software Process Model

This project will follow an agile-waterfall hybrid model, in which the analysis, design, coding, and testing of the project will be completed in an iterative scrum process. The remaining documentation and integration testing will be completed in succession, as shown in the diagram below.



2.2. Roles and Responsibilities

2.2.1. Roles Assignment

Role	Name
Project Manager	Jacob Shirota
Scrum Product Owner	Holiday Pettijohn
Configuration Manager	Holiday Pettijohn
Lead Game Designer	Nicklas Kuo
Lead UI Designer	Caleb Zuniga
UI Designer	Holiday Pettijohn
Application Developer	Caleb Zuniga
Database Designer	Jacob Shirota
Tester	Jacob Shirota

2.2.1.1. The Project Manager will be responsible for the timely delivery of each task and deliverable, as well as sprint schedule management.

2.2.1.2. The Product Owner will be responsible for prioritizing features and requirements. They will also be responsible for deciding which requirements and features to exclude from a sprint, should the need arise.

2.2.1.3. The Configuration Manager will be responsible for owning and maintaining the git repository used for version control and collaboration, as well as the configuration of the Godot engine.

2.2.1.4. The Lead Game Designer will be responsible for the conceptual and mechanical design of Graze, Save, Survive, as well as oversight of the integration of various UI elements.

2.2.1.5. Various UI Designers will be responsible for the graphical and sound elements used to form the front-end interface of Graze, Save, Survive.

2.2.1.6. The Application Developer will be responsible for ensuring that the different components in the game (assets, code, etc) weave together to form a compiling executable application on all target platforms. Nicklas and Jacob will also assist the Application Developer.

2.2.1.7. The Database Designer will be responsible for the design and integration of a database system to the application.

2.2.1.8. The Tester will be responsible for designing, executing, and documenting various tests for each requirement. They will also be responsible for any final quality assurance checks prior to the submission of a deliverable artifact.

2.2.2. Responsibility Assignment Matrix

	J.S.	H. P.	N. K.	C. Z.
Planning/Schedule	R	C	C	C
Requirements Specification	R	R	R	R
Design Description	R	R	R	R
Scrum Prioritized Backlog	A	R	A	A
Version Control Management		R		
Game Design	I	A	R	I
Graphic Design		R	A	
Sound Design		C	C	R
Application Development	A	C	A	R
Database Configuration and Integration	R			C
Requirements Testing	R	A	C	
Quality Assurance	R			
*R - Responsible, A - Accountable, C - Consulted, I - Informed				

2.3. Tools and Techniques

2.3.1. Game Engine: Godot
2.3.2. Database: TBD
2.3.3. Graphic Design: GIMP
2.3.4. Sound Design: Logic Pro
2.3.5. Version Control: GitHub

3. Project Management Plan

3.1. Tasks

3.1.1. Specify Requirements

3.1.1.1. Description: Specify a series of numbered requirements in order to solidify the design of the game.

3.1.1.2. Deliverables and Milestones

3.1.1.2.1. Deliverable: Requirements Specification

3.1.1.3. Resources Needed

3.1.1.3.1. Requirements Specification members

3.1.1.4. Dependencies/Constraints

3.1.1.4.1. Some level of game design needs to be solidified before requirements about game mechanics can be set

3.1.1.5. Risks

3.1.1.5.1. Specified requirements may be over-ambitious - this can be mitigated by prioritizing requirements based on which are essential, important, or desirable. Additionally, tracing the source of the requirements will allow us to see whether they come from the team or from explicit assignment definitions, allowing us to prioritize better.

3.1.1.6. Assignment

3.1.1.6.1. The Lead Game Designer and Scrum Product Owner will be responsible for the elicitation of requirements. The Project Manager will oversee the formalization of these requirements in an SRS document.

3.1.2. Design Gameplay Loop

3.1.2.1. Description: Design general structure of gameplay, based on the roguelite genre. How the game starts and ends is determined, as well as the conditions on which the player achieves ‘victory’.

3.1.2.2. Deliverables and Milestones

3.1.2.2.1. A shared document outlining the gameplay in the repository.

3.1.2.3. Resources Needed

3.1.2.3.1. The lead designer, in consultation with the rest of the team

3.1.2.4. Dependencies/Constraints

3.1.2.4.1. Gameplay must be relatively simple in scope to develop, yet be interesting enough for replayability.

3.1.2.5. Risks

3.1.2.5.1. Testers or team members report the scope of the game is too complex to develop or lacks interest. Simplifying the gameplay will be done, and everyone informed of changes.

3.1.2.6. Assignment

3.1.2.6.1. The Lead Game Designer will outline the high-level structure of gameplay.

3.1.3. Maintain Git Repository

3.1.3.1. Description: Host a Git repository on GitHub such that the project can be compiled into a runnable binary from the contents of the repository alone

3.1.3.2. Deliverables and Milestones

3.1.3.2.1. Milestone: Git repository created and contributors configured

3.1.3.2.2. Milestone: First release/commit where the game compiles

3.1.3.2.3. Deliverable: Compiled binary project ‘release’

3.1.3.3. Resources Needed

3.1.3.3.1. Godot

3.1.3.3.2. GitHub

3.1.3.3.3. The full team

3.1.3.4. Dependencies/Constraints

3.1.3.4.1. Requires all team members to have GitHub accounts

3.1.3.5. Risks

3.1.3.5.1. Merge conflicts may result from uncoordinated edits and commits to files - team communication is necessary to prevent and/or reduce this

3.1.3.6. Assignment

3.1.3.6.1. The Configuration Manager is responsible for maintaining the Git repository.

3.1.4. Create prototype with stand-in assets

3.1.4.1. Description: Successfully compile a build which implements simple mechanics such as opening the main menu, entering a game, and basic HP/damage. This should be able to be compiled on both Windows and Linux at least.

3.1.4.2. Deliverables and Milestones

3.1.4.2.1. Two executable files: Linux and Windows.

3.1.4.3. Resources Needed

3.1.4.3.1. Godot engine

3.1.4.3.2. Stand in assets

3.1.4.4. Dependencies/Constraints

3.1.4.4.1. Requires the gameplay loop, basic combat design, and basic UI design to be solidified

3.1.4.5. Risks

3.1.4.5.1. Overly complicated combat or UI design could make this more difficult than it should be

3.1.4.5.2. Cross platform compiling (Windows for Linux, Linux for Windows, etc) may be difficult and/or hard for the compiler to test - users of a respective system should compile to avoid the need for cross compiling

3.1.4.6. Assignment

3.1.4.6.1. The UI Designer responsible for the procurement and design of graphic assets will work in conjunction with the Application Developer to create an initial prototype with stand-in assets.

3.1.5. Make Progression Persistent - Save System

3.1.5.1. Description: Create a system which allows users to save their progress in the game

3.1.5.2. Deliverables and Milestones

3.1.5.2.1. Milestone: Implement initial save functionality

3.1.5.2.2. Milestone: Implement ability to make multiple save files

3.1.5.2.3. Deliverable: Save file system

3.1.5.3. Resources Needed

3.1.5.3.1. Godot

3.1.5.3.2. Database Software (Likely SQLite or a variant)

3.1.5.4. Dependencies/Constraints

3.1.5.4.1. Need to utilize Godot’s database features to save persistent user data

3.1.5.4.2. Need to have some level of progression designed and implemented beforehand

3.1.5.5. Risks

3.1.5.5.1. Issues with Godot’s database features could impede implementation of the save system

3.1.5.6. Assignment

3.1.5.6.1. The Database Designer will be responsible for the creation of a persistent-save system, and will then work with the Application Developer to integrate the database system into the application.

3.1.6. Secure Assets

3.1.6.1. Description: Create and/or find suitable visual and auditory assets for both the gameplay sprites and user interface elements

3.1.6.2. Deliverables and Milestones

3.1.6.2.1. Milestone: Basic (prototype level) Sprites Added

3.1.6.2.2. Milestone: All static sprites for in-game entities added added and integrated
3.1.6.2.3. Milestone: All sprites for UI added
3.1.6.2.4. Milestone: All animations completed and integrated
3.1.6.2.5. Milestone: Basic (prototype evel) Sounds Added
3.1.6.2.6. Milestone: All appropriate sounds added and integrated
3.1.6.2.7. Deliverable: Sprites and sounds will be present in final release (may not be finalized until then)
3.1.6.3. Resources Needed
3.1.6.3.1. Graphic Design team members
3.1.6.3.2. Pixel Art Editor
3.1.6.4. Dependencies/Constraints
3.1.6.4.1. Need to have some level of prototype in order to assess the suitability of assets in the context of gameplay (color contrast vs the background, etc)
3.1.6.5. Risks
3.1.6.5.1. Creating systems which require too much spritework (lots of enemy types/animations per enemy, etc) may cause this to take longer than anticipated. The Game Design team should plan with reasonable spritework demand in mind.
3.1.7. Design Progression Paradigm
3.1.7.1. Description: Design a system of progression by which the player will grow in the game and decide between possible systems/formats of permanent upgrades which may be obtained after individual ‘runs’ in the roguelite/roguelike format
3.1.7.2. Deliverables and Milestones
3.1.7.2.1. Milestone: Basic gameplay loop established
3.1.7.2.2. Deliverable: This gameplay loop will be present in all builds of the game
3.1.7.3. Resources Needed
3.1.7.3.1. Game Design team members
3.1.7.3.2. Godot engine
3.1.7.4. Dependencies/Constraints
3.1.7.4.1. The persistence of progression will depend on the implementation of a database in order to store what progress has been made in-game
3.1.7.5. Risks
3.1.7.5.1. The gameplay loop may be unengaging or confusing for users - we may not know this until early testing/quality

assurance, so we need to test our game design hands on as soon as possible to avoid/fix this

3.1.7.6. Assignment

3.1.7.6.1. The Lead Game Designer will be primarily responsible for the design of a system of progression. This will be done in consultation with the Database Designer.

3.1.8. Level/Stage Design

3.1.8.1. Description: Design the playing field which the gameplay will take place on, distinct from the user interface

3.1.8.2. Deliverables and Milestones

3.1.8.2.1. Deliverable: Prototype with basic level design during gameplay

3.1.8.2.2. Milestone: Finalize level layout/design

3.1.8.2.3. Deliverable: Final level design along with final game

3.1.8.3. Resources Needed

3.1.8.3.1. Game Design team

3.1.8.3.2. Graphic Design team

3.1.8.3.3. Godot

3.1.8.4. Dependencies and Constraints

3.1.8.4.1. Need an agreed upon framework for combat and the general gameplay loop

3.1.8.4.2. Need some degree of save/database implementation if we want to have more than one stage and/or variants on a stage based on progression

3.1.8.5. Risks

3.1.8.5.1. If this becomes too convoluted and demands too much player interaction, it may interfere with combat design and/or require interaction with combat design

3.1.8.6. Assignment

3.1.8.6.1. The UI Designers and App Developer will be responsible for the design of levels/stages.

3.1.9. Menu Designs

3.1.9.1. Description: Develop an interface for non-gameplay interactions such as save loading, starting the game, and changing settings.

3.1.9.2. Deliverables and Milestones

3.1.9.2.1. Milestone: Basic functionality (main menu, starting game, opening a menu for progression content) implemented

3.1.9.2.2. Deliverable: Prototype must contain at least a rough UI

3.1.9.3. Resources Needed

3.1.9.3.1. Godot

3.1.9.3.2. Game Design Team

3.1.9.4. Dependencies/Constraints

3.1.9.4.1. Need to have some level of sprite/graphical UI design before menu can be implemented

3.1.9.5. Risks

3.1.9.5.1. UI needs to be usable, otherwise it could confuse/deter players. We will tie usability metrics to this element in particular to ensure this does not cause issues for the rest of the game.

3.1.9.6. The UI Designers will be responsible for the oversight of menu design in coordination with the Tester, who will be responsible for the high-level design of menus.

3.1.10. Combat Design

3.1.10.1. Description: Design how the players and the enemies will interact. This includes all moment-to-moment gameplay which involves the player overcoming the challenges enemies create.

3.1.10.2. Deliverables and Milestones

3.1.10.2.1. Milestone: First player-enemy interaction implemented

3.1.10.2.2. Deliverable: Basic combat implemented in first prototype

3.1.10.2.3. Deliverable: Final combat system will be present in final game release

3.1.10.3. Resources Needed

3.1.10.3.1. Godot

3.1.10.3.2. Game Design Team

3.1.10.4. Dependencies/Constraints

3.1.10.4.1. Need to have at least basic sprites before any combat logic can be implemented in-game

3.1.10.5. Risks

3.1.10.5.1. We should avoid making overly complex mechanics which may make features difficult to implement and/or debugging overly difficult

3.1.10.6. Assignment

3.1.10.6.1. The Lead Game Designer and Application Developer will be responsible for the design of combat mechanics.

3.1.11. Playtesting

3.1.11.1. Description: Playing the game in order to test for bugs, glitches, and overall quality and user satisfaction

3.1.11.2. Deliverables and Milestones

3.1.11.2.1. Milestone: Define quantifiable criteria and methodology for playtesting the game

3.1.11.2.2. Deliverable: Initial playtest results for first prototype

3.1.11.3. Resources Needed

3.1.11.3.1. Game Design team

3.1.11.3.2. Quality Assurance team

3.1.11.3.3. External playtesters

3.1.11.3.4. Godot

3.1.11.4. Dependencies/Constraints

3.1.11.4.1. Need to design on game structure before we can define a criteria by which the playtests can be judged

3.1.11.4.2. Need at least a basic prototype before any testing of the game itself can begin

3.1.11.5. Risks

3.1.11.5.1. We will want to contact people outside our group to playtest for feedback, but some people may not reply and/or may be unable to help. We will account for this by having multiple members informed they need to contact people and by contacting people in advance and being flexible with when/where/in what format the tests take place

3.1.11.6. The Tester will be responsible for the design and monitoring of tests, as well as the procurement of volunteer play testers. They will also be responsible for the documentation of testing in two formalized Software Test Documentation documents.

3.2. Timetable

3.2.1. Refer to Gantt chart

4. Additional Material

4.1. Definitions and Abbreviations

4.1.1. “Game” refers to the project

4.1.2. “Gameplay” is the state the game is in from when the player chooses to start the game to when they either pause, quit, or win/lose the game

4.1.3. “Run” refers to an event within the game which consists of the gameplay where the player controls their character

4.1.4. “Mob” or “Npc” refers to a movable entity connected with a sprite interactable during gameplay

4.1.5. “Enemy” is a mob which impedes the player from winning in any way

4.1.6. “Obstacle” refers to any hindrance to the player winning that is not directly tied to an enemy

4.1.7. “Combat” refers to the player interacting with enemies and obstacles to overcome the challenge put forth by the game