NTNU

Kunnskap for en bedre verden

Department of Computer Science

TDT4240 - Project requirements document

Haakon Svane:

MegaGolf

At the opposite end of minigolf

Group:
Group 29

Author:
Haakon Svane (haakohsv)

COTS:
iOS using Swift with SpriteKit, GamePlay Kit and Game Center

Primary quality attribute:
Modifiability

Secondary quality attribute:
Performance
Usability

April 28, 2021

Preface

While there are plans to release this game to App Store, this unlikely to happen before the project deadline. The quality of the game is not expected to exceed my standards, and getting the Apple review team to verify the game can also be a lengthy process. This means that this document may include ideas that are planned to be worked on after the project deadline. These ideas however, might still be significant enough to have impact on the software architecture. This text is intended to address the possibility that the final product may or may not contain all the features mentioned in this document.

Table of Contents

List of Figures

List of Tables

1 Introduction

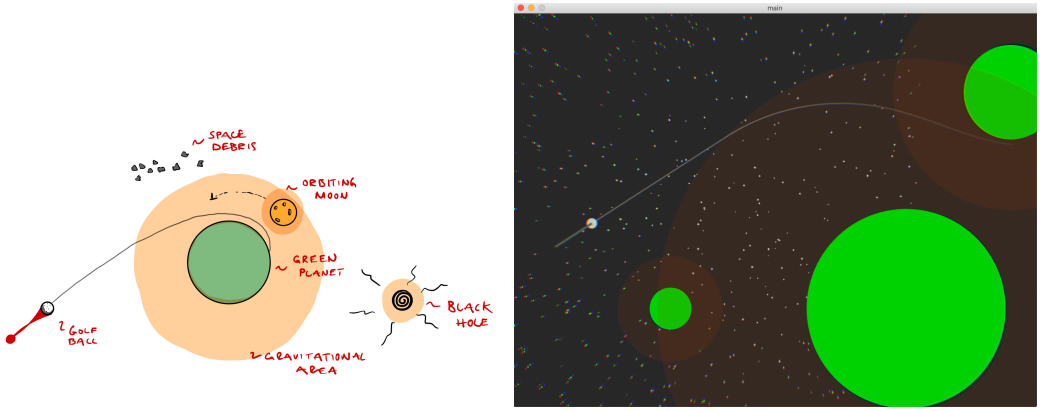
This section provides introductory information about the project, the game concept and the further structuring of this document.

1.1 Description of the project and this phase

This project is a semester project for TDT4240 - Software Architecture. The goal of the project is to design a mobile game (iOS or Android) with an online element to it. A software architecture is to be designed, documented and implemented. The purpose of this document is to act as a basis for the design of an iOS multiplayer game, listing all the necessary requirements needed to fulfil the determined functional and quality attributes.

1.2 Description of the game concept

MegaGolf is a golfing game set in space. It draws inspiration from both mini golf and regular golf. The aim of each level in the game is to make use of gravitational assist from planetary bodies to alter the trajectory of a golf ball, guiding it into a black hole which acts as a hole in regular golf. The game play is depicted in Figure 1a. A small proof of concept (POC) demo was made before project startup to evaluate the core mechanics of the game (gravity assisted trajectories). A screenshot of this demo is shown in Figure 1b.



(a) Illustration of the basic tokens of the game. The orange area around the celestial bodies show the gravitational area of effect.

(b) POC demo of the game to test the gravitational assisted trajectory and camera movement.

Figure 1: Game concept illustrated and demonstrated.

1.2.1 Game mechanics

Each level consists of one or several obstacles and one black hole. An obstacle may be a planet, an orbiting moon, a star or space debris. Planets come in the following categories: field planets, rough planets, water planets and bunker planets. The material that comprise the planets defines its category and their physical properties are similar to that one would see on a regular golf course. Space debris does not have any gravity so they act as deflecting obstacles rather than attracting obstacles. The player spawns some distance away from the goal (black hole) and must navigate to it by launching the golf ball using a drag and release command where the relative position of the finger at release time from the golf ball determines the velocity vector for the ball. The scoring of the game follows normal golfing conventions: each level has an associated par in which the difference between the player score and the level par determines the score awarded for the level. Lower is better.

1.2.2 Menus and modes

The main menu of the game presents a single player and a multiplayer mode. Single player gaming consist of unlockable levels that the player can unlock by completing previous ones. Levels are subgroups of solar systems such that a single solar system contains multiple levels. All levels in a solar system share a similar theme in terms of game play. The multiplayer portion of the game will consist of real-time player vs. player matching. Players will be able to invite friends or play vs random opponents around the world. An online game consists of playing through all levels in a solar system and the winner is the player with the best (lowest) score.

1.3 Structure of the document

This document describes the requirements phase of the project. Section 1 provides a description of the game concept and its mechanics. Section 2 and 3 respectively present the functional and quality requirements set for the project. Section 4 present the constraints on the project as a consequence of the chosen COTS and briefly presents each relevant component. Section 5 presents issues met during the development of the project and Section 6 list the changes that have been made to this document as well as a timetable for the individual contributions made to this document.

2 Functional requirements

The list of functional requirements are presented in Table 1. These serve as a basis for the functionalities that are to be implemented in the project. A priority label is associated with each requirement. The labels and their definitions are

Low The requirement is not essential for the system and its core features, but should be considered for implementation should time allow it.

Medium The requirement is important for the system, but will not serve a as a critical basis in the development of the architecture.

High The requirement is crucial for the system and must be prioritized at the highest degree possible for the design and implementation of the system.

These labels will serve as a basis for devising a strategy on the development of the project. The requirements list also serves as a useful method for error correcting since detecting faulty requirements, for example contradicting ones, at this stage will require a minimum amount of work to correct.

[1] A physical entity is an entity that has physical interactions enabled.
[2] Dynamic sound is defined as music and or sound effects that change according to game-play parameters.

ID	Requirement	Priority
FR1	Interaction between user and system is to be done by user touch control.	High
FR2	User can control the launch velocity of the golf ball by drag and release.	High
FR3	All modes are presented in the main menu.	High
FR4	The game should support single player mode	High
FR4.1	The single player session should display a scoreboard / par info pane.	Medium
FR5	The game should support real time online multiplayer.	High
FR5.1	The multiplayer should utilize Game Center services.	High
FR5.2	A player should be able to invite friends for the multiplayer session.	High
FR5.3	The multiplayer session lobby should auto-fill opponents.	Medium
FR5.4	The multiplayer session should provide a text / symbol chat.	Low
FR5.5	The multiplayer session should provide a scoreboard / par info pane.	Medium
FR6	The game should support a map-maker mode.	Low
FR7	Physical entities[1] should respond to collisions and or gravity.	High
FR8	The game should support dynamic sound. [2]	Medium
FR9	The game should have a settings pane.	Medium
FR9.1	The settings pane should never be more than one view away from the user.	Medium
FR9.2	The settings pane has individual sliders for sound effects and music.	Low
FR10	The game should support achievements through Game Center.	Low

Table 1: Functional requirements for the project. Each requirement has a unique identifier as well as an associated priority label.

3 Quality requirements

Quality requirements target non-functional requirements for the system. These serve as the basis for the development of the system architecture and will play a central role for solving implementation specific problems that may arise. The following subsections present the three chosen quality attributes for the project as well as specific requirements to ensure that these are taken care of.

3.1 Modifiability requirements

The modifiability requirements are meant to aid in ensuring that the system is mutable under minimal cost. The specific requirements target areas of the system that are meant to be modified or built upon. These are determined based on the consideration of *what* parts of the system can change, what is the *likelihood* of the change, *when* the change is made and *who* makes it as well as the *cost* of the change [Bass et al., 2013, p. 117-118]. Tables 2 list the modifiability requirements set for the project.

Identifier	**M1**
Source	Developer
Stimulus	Adding / reworking planets.
Artefacts	Planet interface source code.
Environment	Design time
Response	Implemented and tested in single player & multiplayer.
Measure	Within 30 minutes.

(a)

Identifier	**M2**
Source	Developer
Stimulus	Adding / modifying levels
Artefacts	Level interface source code and editor.
Environment	Design time
Response	Implemented and tested in single player & multiplayer.
Measure	Within 2 hours.

(b)

Identifier	**M3**
Source	Developer / Designer.
Stimulus	Adding / changing textures.
Artefacts	Affected entity source code interface.
Environment	Design time.
Response	Implemented and tested in single player.
Measure	Within 30 minutes.

(c)

Identifier	**M4**
Source	Developer
Stimulus	Modifying component behaviour.
Artefacts	Affected module / component / entity.
Environment	Design time.
Response	Implemented and tested in single player & multiplayer.
Measure	Relevant source code found within 3 minutes. Change does not cause issues outside the affected scope.

(d)

Tables 2: Modifiability requirement scenarios.

3.2 Performance requirements

The performance requirements are meant to aid in ensuring that the system performs to a satisfactory degree. Performance is the ability of the system to meet timing requirements, and these specific requirements must be developed in context of its purpose. The determination of the specific measures stem from both experience, desire and external sources (see: Nielsen [2010]). Tables 3 list the performance requirements set for the project.

Identifier	**P1**
Source	End user
Stimulus	User interface touch event.
Artefacts	UI event handler.
Environment	Run time
Response	User is presented with a system response to touch.
Measure	Within 0.1 seconds.

(a)

Identifier	**P2**
Source	System (internal)
Stimulus	Periodic update
Artefacts	Rendering system
Environment	Normal operations
Response	Data is updated and rendered to screen.
Measure	Throughput of 60 frames per second (median) with 80% stability[3].

(b)

Tables 3: Performance requirement scenarios.

[3]Stability is calculated as the percentage of values in a session that lie within the ±20% range around the specified median value

3.3 Usability requirements

The usability requirements are meant to aid in ensuring a user accomplishes a desired task and that the system provides an adequate amount of user support. Tables 4 list the usability requirements set for the project.

Identifier	**U1**
Source	End user
Stimulus	User interacts with interactibles. [4].
Artefacts	UI / entity event handler
Environment	Run time.
Response	System response to interaction both visually and audibly.
Measure	Less than 5 interactions with non-interactibles for the first 30 minutes of usage.

(a)

Identifier	**U2**
Source	End user
Stimulus	User starts the first level of the game for the first time.
Artefacts	Level management system / first level scene.
Environment	Normal operations.
Response	User is made aware of the basic controls of the game as well as the end goal.
Measure	User will not forget the interactions of the core game play[5] for the following session.

(b)

Identifier	**U3**
Source	End user
Stimulus	User initiates system state transition.
Artefacts	Level management system (Scene transition)
Environment	Normal operations.
Response	For asset loading that will lead to frame stutter, the user is presented with a loading screen.
Measure	User perceives no system halts.

(c)

Tables 4: Usability requirement scenarios.

[4] An intractable is defined as a application element that upon interacted with invokes a system change. This ranges from UI elements to game entities.

[5] Core game play refers to launching the ball to reach the black hole.

4 COTS components and technical constraints

Apple provides large frameworks of functionality that target many specific areas of app usage. A balance must be found such that the research / implementation time of the project is minimal while still conforming to the project requirements. A downside to using high level frameworks is that the developer must conform to the underlying architecture of the framework and all its idioms. This makes implementing software architectures and design patterns more difficult as the designer must bridge the gap between the framework and the desired architecture. The subsections below list and describe the chosen frameworks and some technical constraints that these present to the project.

4.1 iOS and Swift development

The project will be developed using the Swift programming language targeting iOS devices. Due to the project deadline, testing of the game will primarily be done on iPhone XS devices. Due to differing screen aspect ratios and pixel counts between the different iPhone and iPad devices[6], testing them all is a lengthy process and will therefore not be prioritized.

4.2 SpriteKit

SpriteKit [Apple Inc., 2021d] is a general purpose framework for drawing shapes, text and sprites in 2D on iOS devices. It supports lighting effects (shadows, environment lighting and normal mapping), physics simulations (contact detection, collision simulations and forces) and sound simulations (audio playback and positional audio). The source code for the framework is not open, so development using this framework must be guided from Apple's documentation alone. At the core of SpriteKit is the `SKNode` object. This represents any object that can be added to a scene, be it audio, textures, emitters or effects. The system must conform to the idioms and patterns presented by SpriteKit, which may hinder some of its possibilities as mentioned in the beginning of this section. Specifically, initializing a scene and presenting it is done through a provided `GameViewController` object. The framework utilizes the MVC pattern for controlling the scenes [Apple Inc., 2021c]. Utilizing patterns that differ from this therefore requires the design of a subsystem rather than an extension of it. The project will use architectural patterns that are similar to this, so a hybrid case where the Apple's UIViewController will be extended and new functionallity will be added to it.

4.3 GamePlay Kit

GamePlay Kit [Apple Inc., 2021b] is a gaming-centered framework that expands upon SpriteKit to include game-specific services such as AI, pathfinding, and agent behaviour as well as back-end architectural implementations such as an Entities & Components. The source code for GamePlay Kit is, as with SpriteKit, not open to the public. The Entities & Components system provided in GamePlay Kit will be used as a foundational framework in the project. This requires little work to implement with SpriteKit since these two modules work well together.

4.4 Game Center

Game Center is Apple's social gaming network. For developers, it provides an interface for enabling online leaderboards, game achievements and multiplayer functionality such as real-time multiplayer [Apple Inc., 2021a]. Using this interface, the project can incorporate a well supported framework for handling peer-to-peer communications, match finding and online lobby creation. Since this

[6] Apple's senior vice president of Worldwide Marketing Philip Schiller tweeted that the pluralisation "iPhones" is incorrect.

service is provided by Apple, no service sign-up is necessary, and playing with friends is a matter of inviting contacts from your iPhone.

Game Center does not provide a server-client setup, which requires the game to use peer-to-peer (P2P) networking. This has the following consequences:

- **Round-trip time:** Under normal stable network conditions, round-trip time is expected to be better than that of a server-client setup since data is transmitted directly between clients.
- **Instability:** A peer-to-peer network may be more prone to instability since one client with bad connection may cause issues for others as well.[7]
- **Performance:** A server provides computational resources that a networking session can utilize. For a P2P connection, all processing needs to be done by the clients which will impair the client-system performance.

5 Issues

There were no issues faced during the development of this phase.

6 Changes and contributions

Table 5 lists all the changes that was made to this document over time with dates and descriptions. Table 6 lists the individual contributions made to the project at this phase with estimated hours of work.

Date	**Change**	**Description**
2021-03-01	First release	Initial revision of the document.
2021-03-08	Requirements change	Removed requirement **P3**. This should not have been listed in the first place.
2021-04-15	Requirements change	Changed requirement **R9.1** to no longer require a settings pane from every view.
2021-04-15	Requirements change	Rewritten the measures for the usability requirements **U1** - **U3**.
2021-04-15	COTS	Added more information to the COTS section based on feedback.
2021-04-15	Introduction	Added more information to the Introduction section based on feedback.
2021-04-15	Minor changes	Minor changes in various sections based on feedback.

Table 5: Changes made to this document over time.

Name	**Description**	**∼Number of hours**
Haakon Svane	Written the requirements documentation. Researched the functional and quality requirements and made changes to the document.	20

Table 6: Individual contribution of team member(s) for this phase.

[7] While this is dependant on the specific implementation of the P2P network, broadly speaking it still holds.

Bibliography

Apple Inc. Framework: Gamekit, 2021a. https://developer.apple.com/documentation/gamekit/ [Accessed 2021-04-15].

Apple Inc. Framework: Gameplay kit, 2021b. https://developer.apple.com/documentation/gameplaykit [Accessed 2021-04-15].

Apple Inc. Class: Uiviewcontroller, 2021c. https://developer.apple.com/documentation/uikit/uiviewcontroller [Accessed 2021-04-15].

Apple Inc. Framework: Spritekit, 2021d. https://developer.apple.com/documentation/spritekit/ [Accessed 2021-02-25].

Lee Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesly, 3 edition, 2013.

Jakob Nielsen. Website response times, 2010. https://www.nngroup.com/articles/website-response-times/ [Accessed 2021-02-23].