Team Decisions

Your Choice. Your Destination.

Post Mortem of

Sekia

A New World

Choose your own destiny!

Written by Kevin Gisa & Sascha Becker

Version # 1.00

Inhaltsverzeichnis / Table of contents

1 Game Design 3

2 Tasks oft the team members 3

2.1 Kevin Gisa 3

2.2 Sascha Becker 3

2.3 Both 3

3 Used Tools 4

3.1 Gameengine Unity3D 4

3.2 Github 4

3.3 Whatsapp 4

3.4 Skype 4

3.5 Mixamo 4

3.6 Visual Studio 4

4 What worked well 4

4.1 Teamwork 4

4.2 Story 4

4.3 Adjusting the game idea 5

4.4 Learning Unity3D and C# 5

5 What worked not so well 5

5.1 Planning the game 5

5.2 Github usage 5

5.3 Control-structure for events and decisions 5

5.4 Coding Conventions 5

6 What could be the next steps 6

7 Summary 6

1 Game Design

The initial idea of Sekia – A New World was to create a singleplayer RPG where the player can make certain decisions throughout the story. These decisions could vary in the way they affect the story. For example, certain decision would have hardly any impact on the way the game processes, while other decisions would have a huge impact on the story, even resulting in complete new boss fights, the death of major characters or enemies with different attack patterns. However, a combat system could not be implemented due to the time it would take to implement a good combat system. Therefore, we had to change the way decisions would affect the story.
In the actual game, decisions give more options the player can take while talking to certain quest-NPC. For example the player gets a quest where he can decide whether he wants to save a kid or to let it die. If he decides to let the kid die, the people in the town “Kiashi” will not talk to him and thus he is not able to collect all the hints needed to expose Sabura as an enemy. Exposing Sabura as an enemy could lead to him allying with the player if he picks the right choices in the corresponding dialog-sequence.
We also implemented typical RPG features like a Healthbar, an Enembar, which is relatable to Manabars from other RPG´s, an Experiencebar, a corresponding leveling system and a Goldcounter.
Another important feature we could realise is a quest- and a dialogsystem. Like in most RPG´s, completing quests is needed to progress further in the story and the dialogsystem is used to immerse the player into the story and to make decisions.

2 Tasks oft the team members

This chapter describes the main takes assigned to the group members.

2.1 Kevin Gisa

He was the team leader and the contact person for the team’s supervisor. He was also responsible for scripting the quest- and dialogsystem and the UI.

2.2 Sascha Becker

He was responsible for level editing, implementing models, assets and corresponding animations and colliders.

2.3 Both

Both team members came up with the story together. After creating a basic idea, the story got deeper and more complex with each discussion. Both team members were also responsible for bugfixing, documentation and presentations.

3 Used Tools

3.1 Gameengine Unity3D

The team decided to use Unity3D as the gameengine. Unity is extremly popular and therefore there are many video-tutorials or other documentations helping to learn Unity3D really quick. It is also easy to import models, textures and other kind of assets from the Unity-Assetstore.

3.2 Github

The team used Github as the version control system.

3.3 Whatsapp

Whatsapp was used to make arrangements for meetings and other quick questions. Whatsapp makes it possible to contact other team members really quick since many people carry their smartphone with them most of the time.

3.4 Skype

Skype was used to talk to eachother. The „Share-Screen“ function of Skype was also used to visually show the other team member the progression of the game, certain bugs or other visual material needed for discussions.

3.5 Mixamo

Mixamo provides many free character-models and animations. These models and animation were used in the game.

3.6 Visual Studio

Visual studio was used as the IDE for programming C# scripts.

4 What worked well

4.1 Teamwork

The teamwork of the members was working well. Since the team just consisted out of two people it was easy to make arrangements. There were no problems during meetings. Suggestion were analysed critically and the members were understanding.
They also helped each other when the other member had problems, was time-restricted or ill.

4.2 Story

The basic background of the story was already in discussion before the lectures started. After the lecture started and the team members discussed more and more about the story, the story got deeper and more refined with each discussion.

4.3 Adjusting the game idea

When the team realised that the original gameconcept was to complex and would have taken too much time, the members quickly discussed and agreed on dropping features which were too time-intense to implement.

4.4 Learning Unity3D and C#

Thanks to many tutorials and big documentations, getting started in Unity3D went really well, including scripting in C#. When a team member had a problem of comprehension they helped each other to solve the problem.

5 What worked not so well

5.1 Planning the game

Since both team members were pretty new to game development, they underestimated the amount of time it would take to implement their desired features.
Therefore, the team had to drop the idea of implementing a good combat system with their corresponding AI, difficult bossfights and an inventory containing equipment and other items.

5.2 Github usage

Since Github was not used to upload the current state of the project on a regular base, there was an incident where about 20-25 hours of workload was lost due to a HDD-crash.

5.3 Control-structure for events and decisions

A control-structure for checking storyprogression and decisions should have been implemented right at the start of the project. When there were more and more booleans to check, it got really hard to control each of these booleans without overlooking some possiblities.

5.4 Coding Conventions

Both team members used different coding conventions sometimes. They should have agreed on one coding convention at the beginning of the project.

6 What could be the next steps

Since the story is pretty big and deep, it was really hard to immerse the player into the story without the help of good-looking models and animations or a good combat system.
Therefore, with enough time and motivation a good combat system and corresponding bossfights could be implemented. This would help to increase the amount of possibilities decisions could be used to change the game as mentioned earlier.
In addition, an inventory and a gear-system could be implemented. The inventory could include pieces of equipment which boost the abilities of the character and various other items such as healing potions, story-relevant items or simply quest-items.
Also, more quests could be added, letting the player progress further into the story.
Another possible addition could be the inclusion of a changing day-night circle, different weather in different areas and the implementation of good-looking models and animations.

7 Summary

In a summary Sekia – A New World can overall be considered a success. While some initially planned features had to be dropped and some other implemented features have no real purpose yet, the basic idea of creating a game where the player can make decisions which impact the game in different ways has been implemented and works. The teamwork was going well and the team could overcome most of the problems.
The team learned alot about planning a project more realistically, game design and development and the programming with Unity3D and C#.
The consultation with Mr. Miede was really helpful and informative.
The lecture was very interesting and gave the team insight into the process of game design and development.
Therefore the lecture can be recommended without any reservations.