Product Planning
DuoDrive

Pim van den Bogaerdt, pvandenbogaerd, 4215516
Ramin Erfani, rsafarpourerfa, 4205502
Robert Luijendijk, rluijendijk, 4161467
Mourad el Maouchi, melmaouchi, 4204379
Kevin van Nes, kjmvannes, 4020871

May 22, 2014

TI2805 Contextproject, 2013/2014, TU Delft
Group 5, Computer Games
Version 2

Abstract

The product we are making is called DuoDrive, which is a racing game where people have to work together to win. This document will go over the planning that we have made for this product. The planning consists of a high-level product backlog, a roadmap, a list of features sorted according to the MoSCoW method (Beynon-Davies, Mackay, & Tudhope, 2000, p. 210), user stories, and a Definition of Done.

Contents

1 Introduction

The product that we are making is a game. This is not a casual game, but a game where interaction with another player is essential for winning the game. However, winning is not the main goal that we want our players to achieve. We are making this game because we want people to engage in a fun and social activity while they are waiting for something.

The concept of the game is as follows. Essentially, our game is a race game, but the twist is that two people will control the car, instead of the usual one person per car. One person will be in control of the steering wheel and the other person will be in control of the throttle. The game will be played on two different screens. The person in control of the steering wheel has limited view and therefore cannot completely see where the car is heading. What this player can see is how fast the car should go. He/she needs to pass this information to the player who controls the throttle, because the throttling player will not know how fast (or slow) he/she has to throttle. The throttling player can see how the other player has to control the steering wheel and needs to inform him/her about this. The lack of information on both sides forces the two players to cooperate and communicate to reach the finish first (or even at all).

To get this idea from scratch to a final product, we will need a proper planning. In this document, we will first talk about the product itself. Then we will describe the product backlog. Finally, we will give the definition of done. A glossary can be found at the end of the document.

2 Product

In this section we will outline our high-level product backlog and our roadmap.

2.1 High-level product backlog

Our game basically consists of two parts: The *steering player* and the *throttling player*. The steering player has a limited view of the track and will therefore have a hard time controlling the steering wheel. In order to do so adequately, this player has to cooperate with the throttling player, who has a complete overview of the track. However, this player can only accelerate or decelerate. As both players control the same car, they will have to be connected to a network.

Translating the above to a high-level product backlog gives us the following items:

- Create different interfaces for both players.
- Create control events for both players.
- Connect both players through a network.

A backlog containing all features that can be expected, along with the planning of their implementation, is listed in the section below.

2.2 Roadmap

In this section of the report we list the features that we will design. We have divided these features according to the *MoSCoW* model (Beynon-Davies, Mackay, & Tudhope, 2000, p. 210) to give a brief overview of the features and their priorities. The features that must and should be implemented are planned into sprints. The sprint into which they are planned is stated between brackets.

2.2.1 Must-haves

The following features are essential and will be implemented:

- Race track and car (Sprint 1).
- Network server to achieve online player interaction (Sprint 2).
- Different controls for different roles (*steering player* and *throttling player*), including different user interfaces (Sprint 4).
- Road sprites (including deceiving ones) (Sprint 1).
- Steering player has minimal vision of the track (Sprint 1).
- Acceleration player can see an overview of the whole track (Sprint 3).
- Game is playable on devices that run on an Android operating system (Sprint 5).

2.2.2 Should-haves

The following features are not essential to achieve a minimal viable product, but will definitely improve the gameplay:

- Tag teams can match up (Sprint 6).
- Random tracks to remove learning factor (Sprint 5).
- Steering done using a *gyroscope* (Sprint 7).

2.2.3 Could-haves

The following features will only be implemented if time allows us to do so:

- *Ghosts* of other tag teams to see what position you are in.
- Position tracking during the game.
- Several car sprites.
- Track times according to random track length.
- Drop objects on track to harass other players.
- Cross-platform possibilities.

2.2.4 Won't-haves

These features will not be implemented as time will probably not allow us:

- A second mode: ranked mode.

3 Product Backlog

In this section, a number of user stories will be described in regard to the following four points: features, defects, technical improvements, and know-how acquisition. There will also be a section in which we will go over the initial release plan.

3.1 User stories of features

As a player,
Who controls the steer,
I can steer the car

As a player,
Who controls the throttle,
I can determine the speed of the car.

As a player,
Who controls the steering,
I can see when the speed of the car needs adjustment.

As a player,
Who controls the throttle,
I can see when the car needs to steer and where to.

As a player,
Who wants to start a game,
If there is no game available,
I can start a new game.

As a player,
Who wants to start a game,
And there is a game available,
I can join that game.

3.2 User stories of know-how acquisition

As a player,
I know the controls of the game,
because it is explained to me at the beginning of the game,
trough an on-screen explanation.

As a player,
Who drives too fast trough a mud section,
The car slows down,
So that the next time I know I must go slow.

As a player,
Who sees road surface marking on the road heading to the right,
I will go right, but the road does not,
So that the next time I know I cannot trust the road surface marking.

3.3 Initial release plan

For our project we have the following milestones:

- Network implementation finished.
- Gyroscope control.
- Different controls for each player.

We will see these features as milestones for our project in this specific order. The minimal release features are as follows:

- The release must have a track with a car that has limited vision.
- The release must have network capabilities.
- The car must have separate controls.

If these features are not achieved, we cannot release the game.

4 Definition of Done

The last section of this report will explain in detail when the final product can be considered as such. In particular, this section will handle the Definition of Done of a feature, sprint and release.

4.1 Backlog Items

Within the backlog, an item is considered as done, if all the checklist points below are checked and considered to be true:

- Code complete and approved by the lead programmer.
- Code satisfies coding standards.
- Unit tests written and pass.
- Integration system test pass.
- The code is documented.

4.2 Sprint

A sprint is considered done, when all the checklist points below are checked and considered to be true:

- All sprint items considered done.
- Application is tested globally, all unit tests pass.
- Tests pass in the continuous integration system.
- User tests pass.

4.3 Releases

A release is considered done, when all the checklist points below are checked and considered to be true:

- The product should pass all unit tests.
- Integration system test pass.
- Interface looks as the product owner demanded.
- (End-)user tests pass.
- Code documented and satisfies coding standards.

Let it be clear that all of the above section cannot be done if the continues integration tests fail.

The main version of the product must apply to the sprint subsection of the Definition of Done. This version is on the ”master” *branch* in *Git*. The other versions (on other branches) do not have to apply to this, but at the moment the are *merged* into the main branch, they apply to the Definition of Done of the main version.

5 Glossary

Branch
A branch in Git is a separate environment where people can work on part of the code. When this part is done, it can be *merged* into another branch.

Ghosts
You see another player's shape (ghost) in a transparent color. You can not collide with the ghost. It can be used to show how well your opponents are doing.

Git
Git is a version control system. Among other things, this means that multiple people can work on the same code simultaneously, and a history of modifications is saved for reference.

Gyroscope
Device used for measuring or maintaining orientation, as built in smartphones, for example.

Merge
In Git, merging a branch X into another branch Y means applying the modifications of X to Y.

MoSCoW (Beynon-Davies, Mackay, & Tudhope, 2000, p. 210)
Technique to reach a common understanding on the importance of each requirement. 'M' stands for 'Must have', 'S' stands for 'Should have', 'C' stands for 'Could have', 'W' stands for 'Won't have'.

Steering Player
The player that can only control the steering wheel (to the left or to the right). This player can not accelerate or decelerate.

Throttling Player
The player that controls the throttle and can only accelerate or decelerate. This player cannot control the steering wheel.

References

Beynon-Davies, P., Mackay, H, & Tudhope, D. (2000). 'It's lots of bits of paper and ticks and post-it notes and things . . .': a case study of a rapid application development project. *Information Systems Journal*, 10(3)195-216. doi:10.1046/j.1365-2575.2000.00080.x