Project plan+study diary
JangliGame
version 1.5

TUT	Pervasive Computing	TIE-21106 Software Engineering Methodology
Author: <responsible person>		Printed: 11.03.2018 19:43
Distribution: <list of people who will have access to this doc>		
Document status: draft		Modified: 11.03.2018 19:43

VERSION HISTORY

Version	Date	Authors	Explanation (modifications)
1.0	23.1.2018	Daniel N	End of Sprint 0 Version
1.1	16.2.2018	Daniel N	End of Sprint 1 Version
1.2	8.3.2018	Daniel N	End of Sprint 2 Version

TABLE OF CONTENTS

1.   PROJECT RESOURCES

This chapter holds the project resources.

1.1   Personnel

SCRUM MASTER – Atte Lamminsalo
SCUM MASTER – Daniel Natri
SCAM MASTER and COURT COMPOSER – Leevi Kulju
MASTER BLASTER – Henri Laakso

Contact information:
atte.lamminsalo@student.tut.fi
daniel.natri@student.tut.fi
leevi.kulju@student.tut.fi
henri.m.laakso@student.tut.fi

1.2   Process description

Our project goals are to make a simple game of Jungle Hunt. Our sprints end on Sunday. Our main virtual communication channel is Telegram.

Below is our rough schedule for the project requirements per sprint.
We also plan to meet every Thursday at 18:00. In these meetings we will agree on who will do and what during next week. Every time we start implementing a new user story, we will divide it to tasks, if not already done. Whenever possible, group members will code together together. Project will be split into four sprints. The first sprint starts on 8.1. and the fourth sprint ends on 29.4. Pre-determined user stories will do for milestones.

Our team assigns themselves tasks from Agilefant based on what can be done at the moment. If a user story is clearly a package one person can handle by themselves, the team member can assign the whole story to themselves as well. The SCRUM master checks to see if the tasks are evenly divided and possibly forces tasks on lazy team members.

Unfinished user stories are moved to the backlog and usually into the next sprint.
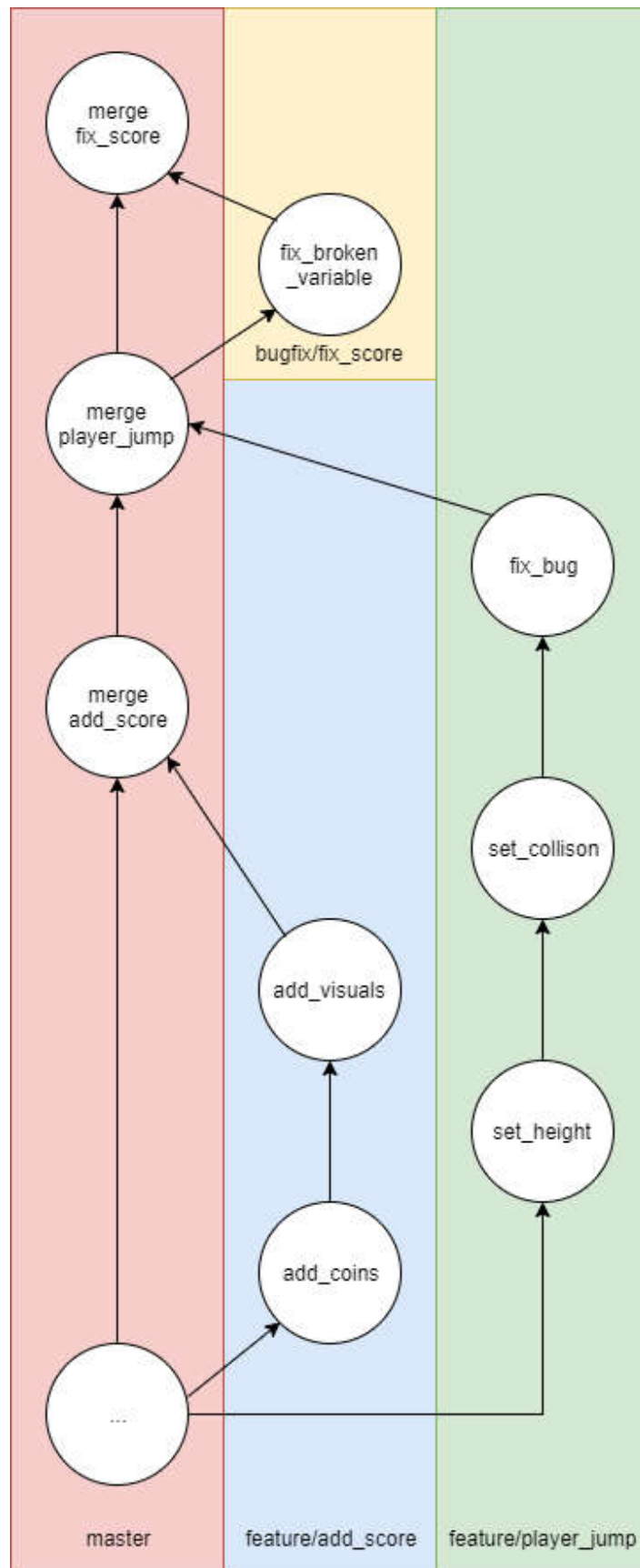
1.3   Tools and technologies

1.3.1 Agilefant

URL: https://app.agilefant.com/TTY-TIE/product/376289/tree

1.3.2 Version control

We are following a master-feature version control schema on our project. The master branch is supposed to always be a working copy of the program. Developers create feature branches for new features, for example *feature/add_score_system* that is merged onto the master branch once fully implemented. This is done in Gitlab via merge request. The merge request is inspected by a team member, and only merged once it has been accepted. This ensures that development is never halted due to a broken copy of the program. If a broken version is merged into the master branch, a quick bugfix branch, e.g. *bugfix/fix_score_system* should be implemented as soon as possible. This is visualized in picture 1.

If new versions of tools we are using come up, we only update the master branch once it is clearly evident that the updates do not break the project.

Picture 1. Master-feature schema

Table 1.1: Tools used in the project.

Purpose	Tool	Contact person
Documentation	MS Word (word processing) office.microsoft.com	D.N.
	UMLet	L.K.
Communication	Telegram□HYPERLINK "http://www.mirc.com"□HYPERLINK "http://www.mirc.com"□HYPERLINK "http://www.mirc.com"□HYPERLINK "http://www.mirc.com"□HYPERLINK "http://www.mirc.com"	A. L.
Version management	Git	A.L.
Game engine	Unity3D	H.L.

2. STUDY DIARY

This chapter holds your journal of lessons learned during the course. That is, **more detailed analysis of previous Sprint's contents**.

2.1 Sprint 1 (every sprint as a section)

2.1.1 What went well

We got the skeleton for the Unity project and learned to use the game engine. Team members were introduced to the toolchain. We were also able to create a basic game with a controllable player. Our group meetings were productive. We had fun. :)

2.1.2 What difficulties you had

We had understood the scrum stories a bit wrong. We underestimated the required time for learning to use the tools. The risks we found were not accurate enough.

2.1.3 What were the main learnings

We learned that it is quite difficult to estimate the required time to complete a task. Weekly group meetings helped in continuous progression in project development and allowed us to share knowledge and solve problems as a group.

2.1.4 What did you decide to change for the next sprint

We will improve our user stories and do more planning for our project. We will also do more risk management and continue to evaluate current risks. We also updated our risk management to be more accurate. Version control and process description got updated. We decided to continue user stories that we didn't complete in sprint 1.

2.2   Sprint 2

2.2.1 What went well

We managed to complete almost all the tasks on the list. Some of the stories were harder than the others. However, we did not have time to merge all the stories to the master branch.

2.2.2 What difficulties you had

We spent a lot of time in testing the components of the program. Some persons from our group had time off because of sickness and stress. There might have been some overestimation in work quality and resources. Merging same unity scenes was difficult and it took more time than we expected and caused a lot of "*mergeles!*".

2.2.3 What were the main learnings

We learned how to use unity and project tools better. Also implementing stories became clearer. Seems like every sprint we can estimate the required time better, but we should always overestimate the required time for a task. We talked about implementing a test script using Gitlab CI, but we opted not to due to time constraints.

2.2.4 What did you decide to change for the next sprint

We will pick tasks more wisely so we can ensure the completion of the tasks as well as merge the changes in time. We also have to work more throughout the sprint instead of just half of the sprint to maximize code quality. We

3. RISK MANAGEMENT PLAN

CHANGES FOR SPRINT 2: The risks were re-evaluated and adjusted to be more realistic. No changes to risk management plan

The most probable and impacting risk that can happen during the process is insufficient planning which leads to increased workload. Other possible risks and their consequences are described below.

Table 4.1: Project risks.

Risk ID	Description	Proba-bility	Im-pact
1	Short term absence	3	1
2	Insufficient planning	3	2
3	Long term absence	1	2
4	Changed customer requirements	3	2
5	Communication problems	1	1
6	Overconfidence in own skills	3	2

3.1.1 Short term absence of one person

It is very likely that someone will get sick during the course. Impact of this is directly dependant on the length and severity of absence. In some cases, short term absence can be negated by online meetings if possible. We can minimize the damage these absences cause by not scheduling our work to last days of sprints and doing all the work in time.

3.1.2 Insufficient planning

It is very likely to have problems caused by insufficient planning because we are learning new skills and our evaluation skills regarding the tasks may not be adequate. Severity of these problems can be anything from total catastrophe to small inconvenience. Most of these problems can be avoided by allocating enough time for planning and sticking to said plans.

Insufficient planning covers all possible problems that might occur regarding required time for certain task. This covers inability to learn and use new technologies according to plan or general extra time spent over estimated.

3.1.3 Long term absence

This is almost the same as short term absence, but this also includes someone of the team quitting. It is very likely that someone will get sick during the course, but not so likely that someone will quit the course. Impact of this is increased workload on the other remaining colleagues. In some cases long term absence can be negated by online meetings if possible. We can minimize the damage these absences cause by not scheduling our work to last days of sprints and doing all the work in time. Also workload should be somewhat equal so no one gets burned out.

3.1.4 Changed customer requirements

Customer might change his requirements which leads to more work and even possibly failure to complete the project. This can be avoided by constant communication with customer and ensuring our product is developed in the right direction.

3.1.5 Communication problems

We might be unable to reach one person of the group for some time which causes some inconveniences for the whole group but these inconveniences should be minimal. Probability for this to happen is very low.

3.1.6 Overconfidence in own skills

A developer can easily overestimate his or her skills technology-wise. This can cause speed bumps in development and serious time loss. The amount of overconfidence can vary in size which causes problems ranging from negligible to problems in delivering user stories.