Sprint Review for Week 5

Context Project: Computer Games
Group: 4
Based on pair programming.

User Story Numbers:	**Task:** Ordered on highest priority first and lowest priority last	**Task Assigned To:** [Responsible] ([Partner])	**Estimated Effort per Task:** Effort prediction in total amount of hours spent by all team members:	**Actual Effort:** in total hours by all team members:	**Done:** Yes/No	**Notes:**
1	Create a default style for all the GUI elements	Nick (Jurgen)	12	7	yes	Has to be refactored. Functioning but sloppy code.
2	Make a stroll and be able to encounter events	Ben (Nick)	12	25	partial	Works, but because of bad programming practices to get a working demo a lot of refactoring had to be done. Resulted in not properly tested code.
	Let the player receive rewards after a stroll has ended		3	1	yes	
3	Make a playable event*	Jean (Ben, Martijn)	36	43	partial	Not tested, because refactoring is needed before we can test.
4	Optimize the GUI for different devices	Jurgen (Martijn)	12	36	yes	Currently only landscape. Had a lot of issues with viewports. More info in *problems encountered*.
5	Define the general concept of the game	Jean (Ben)	18	4	no	Did not spend enough time on this, thus not done.
--	Create a name for the game.		3	1	yes	Name = Super Starfish Mania
-	Create an Assets class	Martijn (Nick)	12	8	yes	
-	Update the emergent architecture design	Jurgen (Ben)	15	5	yes	

* We did not take in account the time it would take to come up with an architecture. The event itself did not take as much time as we estimated but we added the hours for the architecture design here. We'll come back on this topic in the retrospective.

Extra Task:	Task description:	Time spend	Notes
I	Switchable screens	10	Screens had to be made easily switchable. A simple call to setscreen now makes this possible.
II	Refactoring timer	12	Timer and Timekeeper are very enforcing on how to use them. Refactored a lot but still needs more refactoring.
III	Adding Game logic	12	Game logic and screen display have to be independent.
IV	Creating Event Architecture	16	The event architecture is a base on which we can make new events.
V	Merging of branches	4	Merging branches takes time, since each developer made many changes on his own. Some cleanup was needed.

User Stories	
User Story Number:	**User Story:**
1	As a developer, I want to be able to easily add a new button that has the same style as the rest of the game.
2	As a user, I want to be able to go on a stroll and play events during this stroll.
	As a user, I want to receive rewards when I successfully complete an event during a stroll.
3	As a user, I want to play an event.
4	As a user, I want to be able to play the game on different devices without problems.
5	As a user, I want to play a game with a well thought out concept.

Sprint Retrospective:

Main problems encountered:

- Adding and deleting Timers and TimerTasks was met with a lot of difficulty. The biggest problem here was '*Concurrent modification exception*'. An example is unsubscribing while another Timer were still being '*Ticked'*. Trying to delete something from a list that is still being iterated gave the exception. Thus we had to refactor our code.
- Because we are using a framework some things are not always as clear as we expected. Properly *disposing* objects in order to prevent memory waste/leaks was always as simple.
- Our code has to be refactored a lot since design patterns are not always properly used. Eventually we get the code as we want, but never in the first try. This takes more time from the progress we make.

Major differencing in expected time and actual time:

- The GUI skin has been created, but can be made better. Currently it really is sloppy. The reason we haven't spend the specified amount of time on it is because other tasks took longer than expected.
- GUI optimization for multiple devices is done, but took a long time. Since we work using LibGDX we first had to figure out how to do things the LibGDX way. We tried multiple strategies, like multiple viewports and containers. The problem was that any implementation we tried to make eventually resulted in another problem problem to fix. Trying to solve that problem would create new problems and so on. The current implementation is based on the things that did work after a lot of trial and error and might not be the best implementation.
- Creating the strolls is significantly longer than estimated since we first made a running version as soon as possible, for the demo, but later had to refactor. The GUI and the logic for the strolls and events had to be decoupled so it could be tested.
- We didn't get to define the concept for the game simply because we did not have the time to get to this part. We wanted to have a clear document of what we are going to build, but will now just build this as we go on with the project.
- The Emergent Architecture Design document is something that can only be 'updated' with the things we have implemented. We wanted to complete the document as much as possible (and create code based on the EAD), but we found that we have to refactor a lot. Updating the EAD will now be done once we have something functioning and should not take too long.
- We also added a table of '*extra tasks*'. These are tasks that can't be put within the original sprint plan tasks. The first *sprint improvement* will discuss this further.

Positive improvements:

- This week we had no problems with git. Although the merging and cleanup took some time, we did not have to fix anything. Since all members know how to use git now, working together is easier.
- Because this week a playable spike had to be completed, we actually got to see some progress. Actually seeing the game taking a tangible form is really satisfying.

Improvements for the next sprint:

In order to improve for the next sprint we want to:

- Improve the tasks description. The tasks, as we defined them at this moment, are too broad. We found that it is hard to make an estimation for the actual effort when many tasks 'intersect' with each other. The next weeks we will make tasks as specific as possible, but leave room for variation. This means that in the sprint review, we add 'subtasks' if these are applicable for tasks in the original sprintplan;

- Make better use of the daily meetings and actually do them ‘as supposed’. Every morning we will discuss what every person has done as well as what he will do for that day. This will hopefully make it clear when we are going to lack time for the sprint. If needed we can do more at home to compensate;
- We want to better estimate the time a task takes. Even though we said this before, it still remains difficult to make predictions on the effort a task will take. This is because we have too broadly defined tasks as well as an overestimation of our abilities. Certain tasks seem simpler than they are. This is usually because we lack the knowledge of how the LibGDX framework works or how to get components to work together;
- Currently we work individually at first and at a later point inspect the code together. We want to work together more from the start, so we can quickly detect if a certain implementation can be improved or refactored. This prevents us from having to make lots of changes at a later point;
- At the time testing is something we do while also writing the code, not actually making a test class and then writing the actual code. If we first write what we expect, we can quickly see if something is not going to work.