Sprint Review for Week 4

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 scales based on lightest task	**Actual Effort:** in total hours by all team members:	**Done:** Yes/No	**Notes:**
-	Product Planning (final)	Jean (Martijn)	2	16	Yes	
-	Extend EAD with design documents that we create this week	Ben (Jurgen)	3	0	No	Spend too much time on Timer. Also need to think out the way we want dependencies to be accessed, after that we can make interaction diagrams, UML etc.
1	Timer keeps running when the game is closed or minimized.	Jurgen (Ben)	3	35	Yes	Saved in preferences instead of actually running as a service. Timer 'created' from preferences on app start.
	Timer keeps running when the phone is turned off or rebooted.				Yes	
2	Create a 'Stroll' screen that has to display a timer.	Nick (Jean)	1	14	Yes	Unfortunately due to dependency issues this has only been partly completed. Due to these issues the time which could not be put into these tasks has been put into tasks which were planned for the near future. These tasks will be completed in the near future. Probably within 2 days.
	After the timer hits 0, the screen is changed to a reward screen.				No	
	The reward screen has a button to return to the main menu.				No	
3	On random intervals during a stroll, an event occurs.	Jean (Nick)	2		Partly	
	Whenever an event occurs, the user is notified by a pop up.				No	
	The player can acknowledge the notification by pressing the "OK" button.				No	
	When the "OK" button is pressed, the notification disappears and the player returns to his/her stroll.				No	
-	AssetManager	Martijn (Nick)	1	5	No	Researched the different possibilities. Team decisions not yet made (because of public holiday).
4	Screens should be easily switchable and as independent as possible.	Martijn (Nick)	2	2	Partly	We want to save the screens in memory. Currently we always create new Objects but we want to refactor this to be more efficient.
	Add a button to the main menu to go on a stroll (iff the player has waited long enough)				Yes	

User Stories	
User Story Number:	**User Story:**
1	As a player, I want to be able to close or minimise my application, so that I can use my phone for other things and still go on a stroll every hour.
	As a player, I want to be able to turn off or reboot my phone while the timer keeps running, so that I can still go on a stroll every hour.
2	As a player, I want to be able to see how much time I still have left on my stroll.
	As a player, I want to be able to see which rewards I have earned after a stroll.
	As a player, I want to be able to to return to the main menu, after I have accepted my rewards.
3	As a player, I want to events to randomly occur while I am on a stroll.
	As a player, I want to be notified when such an event occurs.
	As a player, I want to be able to acknowledge the event.
	As a player, I want to be able to go back to my stroll when the event has been completed.
4	As a developer, I want to easily be able to implement and add new screens, so that the implementation of the project will run more smoothly.
	As a player, I want to be able to press a button on the main menu, so that I can go on a stroll when a stroll is available.

Sprint Retrospective:

Main problems encountered:

- Because of holidays, we had less schedule days to work at the university. This made working as a team less effective. We did keep the holidays in mind, but had a hard time getting together on those free days. Individually we did sufficient, but having set hours to work together makes discussing problems a lot easier.
- We had some difficulties with git merges. Either by accepting certain files or parts of files, the resulting merge sometimes was different than we expected. This was confusing and took a long time to fix.
- Sometimes one member had problems with either git or other applications. When other team members help out they can’t do their own tasks. This reduces efficiency for both members.
- The Timer was required for the stroll and the events. This meant that there was a dependency for the Stroll and Event tasks. Before the Timer was completed the members that had to make the Stroll and Events couldn’t start their work. This created a problem to do parallel work. This gap was eventually filled with doing separate (non planned) tasks.
- We underestimated the time testing and checking the code took. Testing was hard to get working because of the LibGDX framework that creates certain required static classes. Eventually this was solved by using a helper class found online. We also made testing more accessible by using protected methods and variables instead of private.
  Getting all checkstyle errors out of the code also was a pain, but eventually worked out. It just took a lot longer than we expected at first since fixing one error sometimes led to new errors appearing.

Positive improvements:

- No particular improvements compared to previous weeks. We did a good job together and finally got to do some actual coding instead of only planning and discussing.
- Members did other (useful) tasks that were not in the sprint plan when no other tasks assigned to them were available. Working on the same part of code with 3 people is not efficient, but having one person look into the framework or other necessary parts of the game for the coming weeks also gives an idea of how difficult/useful certain features of the game are.

Improvements for the next sprint:

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

- We want assign tasks better in order to reduce the ‘idle’ time that is created by having some depended tasks (for task B we first need task A to be completed). This means that someone should have some ‘independent’ tasks when he also has dependent tasks assigned. Ideally would be that the independent tasks are as much or more than the effort required for the ‘dependency’. This would ensure there is no idle time.
- Everyone should look into Git branching and merging. By reducing (or nullifying) the amount errors made in merging we can focus more on creating new code instead of fixing these errors. More time spent coding is more effort in the actual game.