Product Plan

Context: Health Informatics

Group: HI1 a.k.a. Geen Naam (Group Number 4)

04-05-2016

Group Members:

Name:	NetID:	Studentnumber:
Jonas Dieterich	jdieterich	4317424
Maiko Goudriaan	maikogoudriaan	4302125
Ruben Meeuwissen	rmeeuwissen	4287150
Louis Sikkes	lsikkes	4298977
Thijmen Jaspers Focks	thijmendewilde	4158393

Table of Content

Introduction

In the health informatics context we are working on a solution to help people with various sorts of psychoses. To resolve the problem we are using virtual reality therapy. Patients get to interact with a virtual world set in a shopping street by using virtual reality glasses. The therapist gets to track the patient's movements in the virtual world using a 2D map representation that displays all characters, objects and map boundaries. The therapist is able to let the characters interact with the patient by triggering actions such as talking, moving (e.g. following the patient), interacting with other characters/objects in the world and dynamic emotions.
With this application it is possible to analyze the patient’s behaviour and treat the patient’s condition during exposure to the virtual world.

In this document we will make a general plan for the product. We start off with a MoSCoW plan that describes which features we must have, should have, could have and won’t have. Secondly, we will provide the milestones for each week which describe what features of the product should be done by when. Then we provide a roadmap that delves a little deeper into what the tasks are due per week. Finally we end with the definition of done. The appendix contains the user stories describing why and what actions are implemented and how they are performed.

Product

High-level Product Backlog

Must have:

Non-functional:

- General
    - Fluent motion (60fps using PC) & (90fps using Oculus Rift)
    - Processing time <= 1ms
    - Difference of occurrences in VR and on 2D map (<= 0.2s) -> responsiveness
    - Scalability (adding actions, adding large amounts of characters, adding group behaviour, monitoring features e.g. heartbeat)
    - Thorough documentation
    - List of prescribed tools used for code maintenance
    - Patient main focus, not 2D map (helping tool)
    - Short manual for users
- Unity 3D World
    - Use C# for coding
    - Unity 3D (5.3.4 or 5.4b)

Unity profiler for performance related evaluation

    - .NET 3.5
- GUI - VS
    - Use C# for coding
    - Use XAML
    - Use WPF
    - Use the MVVM design pattern
    - .NET 4.5 (or higher)

Functional:

- Show world objects:
    - Buildings
    - Trees
    - TV screen
    - Benches
    - Police cars
    - (more can be defined later)
- Show subsection of the map
- Enlarged icons (for clarity)
- Map needs to be movable
- Objects are clickable and can perform 2-click actions (or 1-click, hover -> click):
    - Change emotions
    - Looking at the patient

        - Following the patient
    - Dragging/moving icons (to trigger walk/follow)
    - Field of vision for patient to show where the patient is looking
    - Network communication (minimal delay between GUI and VR computer)

Should have:

Non-functional:

- Logging of events/actions taking place
- Code should be open and it should be easy to add new actions and extend the current map with additional features

Functional:

- Perform additional 2-click actions (or 1-click, hover -> click):
    - Interact with world objects (sit on bench, etc.)
    - Change state of world objects (change TV screen text, turn on police car sirens, etc)
    - Trigger speech of characters
- All character group/list
- Current emotion of character displayed next to their portrait
- Group actions
    - Create / disband group
    - Move group to location
    - Change emotion of group
    - Group looks at patient
    - Trigger speech of group
- Hover on setting bar, to display essential setting options (for quick access)

Could have:

Non-functional:

- Touchscreen ready (adaptable for use with tablet)

Functional:

- Mini-map of the environment
- Keyboard shortcuts
- Search objects/people
- Object labelling
- Text-to-speech announcements (in entire VR World) for triggering speech of a group
- Map performance monitor
- Predefined walking routes
- Predefined actions/ list of actions to carry out for characters

Won’t have:

Non-functional:

- Avoid use of external packages due to compatibility & licensing issues

Functional:

- Voice control
- Pause / stop button (we should just detect when the environment does this and process that)
- Additional features like graphs, a notepad, etc.
- Varying map start-up settings using init window depending on kind of psychoses (with default)

Initial Release Plan

In order to ultimately release our final product we will use an overview of what has to be done each week (expressed in milestones).

Week	**Milestones**
1	Set up of the project.
2	The user has a global overview of the map layout.
3	The map displays virtual characters and virtual objects. Basic controls are available.
4	Communication between VR world and the map is established. On start up of Unity VR world our map displays the initial state.
5	The map is dynamic, hence will update frequently and display the current state of the VR world.
6	Objects are clickable and show available actions. Advanced map controls are available.
7	Available must have actions can be executed when clicked. The user can manually move the characters on the map.
8	Available should have actions can be executed when clicked. Characters are able to follow the patient.
9	It’s possible to create and disband groups. Group actions can be executed.
10	Chosen could haves are implemented. Final product is presented

Roadmap

This is an overview of what tasks need to be done throughout the project. Firstly, a static map is created, which will create the base for our transition to the dynamic and interactive map. The first five weeks are clearly defined, however as the project develops new insights will emerge as new input from CleVR can steer our product in a slightly different direction.

Week	**Task**
1	Setup initial project Divide roles and responsibilities
2	Gather requirements from CleVR Define the project vision Make a project plan Make an architecture design Make the initial view of the map (prototype)
3	Create static map that shows: - Virtual characters - Virtual objects Getting familiar with the given environment Create basic controls for the following aspects: - Move the map using key inputs - A control menu is available
4	Process IO from VR world (understanding available actions, list of interactive characters and objects, map boundaries, layout of map) Start network communication On start up of Unity VR world our map displays the initial state Create a logging system.
5	The map responds to the input of the environment for the following aspects: - Movement of moveable objects - Field of vision of the patient - Emotions of characters Let the logger logs actions
6	Make objects and characters clickable and show their corresponding actions. Make the UI for action menus of the characters and objects. Make the map scrollable.
7	Make actions for the following things: - Trigger speech of characters - Change character emotions. - Let characters walk a drawn path
8	Make an action so that characters can follow each other. Implementing “Should have” actions (list extended throughout project)

9	Make group selection and disbanding Make group actions
10	Make final version of documents Write a manual for the user Implementing “Could have” actions (list extended throughout project)

Definition of Done

Working on a product can continue forever. You can always improve the product. Also you can deliver a product that isn't finished. To guarantee that the product has the right quality we create a definition of done (d.o.d.). We can divide the d.o.d. in three parts: feature, sprint and demo. For a feature to be done the following list should apply[1].

1. The code should not have any compile errors.
2. The code should not violate any errors from the static analysis tools (e.g. checkstyle).
3. The feature should be implemented conform to the defined requirements.
4. The code should accept the UAT (user acceptance test).
5. All code should have understandable comments.
6. The new code should have at least 75% test coverage.
7. The old tests should still succeed, or correctly edited according to the new code.
    a. If it is a minor (1 class is affected) change one person should have reviewed it.
    b. If it is a larger change (2 classes or more affected) two persons should have reviewed it.
8. Relevant documents (e.g. UML or diagrams) are created, when needed.
9. Trello and the retrospective are updated according to the achieved feature.

To know when a sprint is done. This is automatically achieved when the time is over. To determine if it was a successful sprint we can look at the following list.

1. All must haves should been implemented for the sprint.
2. All should haves should been implemented for the sprint.
3. All features should be done specified in the d.o.d. for features.
4. Required documents are created and handed in.
5. No critical errors or bugs are present at the end of the sprint.

Finally we determine when we can release the product to the customers. This means we are satisfied with the product and think that the customers are also satisfied with the product.

1. All must haves should be implemented.
2. All should haves should be implemented.
3. If a must or should have has not been implemented a clear explanation is given, with an appropriate solution.
4. The product is accepted by all the programmers.
5. The product is accepted by CleVR.
6. The product is well documented.
7. The product does not have critical errors or bugs.

[1] Waters, K. (2007, July 26). Definition of DONE! 10 Point Checklist
. Retrieved from http://www.allaboutagile.com/definition-of-done-10-point-checklist/

Glossary

VR - Virtual reality
UI - User interface
IO - Input output stream
CleVR - Stakeholder

Appendix

User Stories

Note: “All characters/people/persons” and “everyone” shall be interpreted as all characters except the patient present in the environment.
The users stories are given in straight letters, while the use-case is given in italic.

See the Virtual World:
As a user, I want to be able to see the virtual world, in order to interact with it.
By putting on the virtual reality goggles (Oculus Rift) the user is exposed to the virtual world and can interact with it.

Interact with the 2-D map:
As a user, I want to be able to interact with the 2D map.
In order to interact with the 2D map, the user needs to access the PC that runs the GUI, which is in turn connected to the PC running the virtual world (via network), that is in turn connected to the VR goggles for displaying the VR world.

Show actions:
As a user, I want to be able to see which actions characters can perform, so that I can change the environment.
Interactive characters or objects can be clicked on (main icon) to list their options. These will be displayed in a circular manner around the icon.

Walking:
As a user, I want to be able to let characters walk to a certain destination, so that I can manipulate the environment.
When you click on a person and draw a path to a certain location, without moving through solid objects/map boundaries, the person should walk the drawn route to the given destination.

Walking cycles:
As a user, I want to be able to draw cyclic paths that enable a character to move repeatedly on a certain route.
When clicking on a person and drawing a path that represents a closed cycle, that character will move repeatedly on that route until instructed to stop (by clicking on the person and terminating the action).

Walking into wall:
As a user, I want to be able to stop a character at an unmovable object, so that I prevent a collision.
When you click on a person and drag them to a certain location, moving through a solid object like a wall that cannot be interacted with, the person should walk until encountering the object and stop.

Following a person:
As a user, I want to be able to let one character follow another throughout the environment.
When clicking a person p1 and dragging the icon onto another person p2, releasing the mouse button will cause p1 to follow p2 around the map.

Stop following a person:
As a user, I want to be able to stop a character from following another character, so that i can influence the patient's experience.
When a person p1 is following another person p2, clicking on the main icon of p1 will list an action "stop following p2", which after being clicked will cause p1 to stop following p2.

Interaction with environment:
As a user, I want to be able to let the characters interact with the environment, so that the environment becomes part of the experience.
When you click on a person and drag the icon onto an environment icon (e.g. a bench icon), the person will interact with object (e.g. the bench by walking towards it and taking a seat).

Changing emotion of person:
As a user, I want to be able to change the emotion of the characters, so that I can influence the patient's experience.
By clicking on the emotion icon (top right of main icon), a list of emotions will be displayed. The emotion selected by a click will be the emotion expressed by the person.

Selecting a group:
As a user, I want to create groups, so that I can move a group of people in a certain direction.
Clicking on the 2D map (not on an icon) and dragging the mouse into a certain direction will create a selection window, all characters (not objects) that fall into this window will be selected after mouse release.

Triggering group actions:
As a user, I want to let groups of characters perform certain actions, so that i can influence the patient's experience.
The 2D map will have a sidebar (on the left of the map) containing a group icon that lists all groups. Clicking on the group icon will list all groups, by clicking on one of the groups, all possible group actions and an emotion icon will be displayed. One of the actions can be triggered by clicking on it.

Changing emotion of group
As a user, I want to change the general mood of the selected group, so that i can influence the patient's experience.
The 2D map will have a sidebar (on the left of the map) containing a group icon that lists all groups. Clicking on the group icon will list all groups, by clicking on one of the groups, all possible group actions as well as an emotion icon will be displayed. By clicking on the emotion icon and selecting an emotion all group members will adopt the selected emotion.

Changing emotion of all people in environment
As a user, I want to change the general mood of all the characters in the environment, so that i can influence the patient's experience.
The 2D map will have a sidebar (on the left of the map) containing a group icon that lists all groups. One main group called "All" will contain all characters of the map. By clicking on the group, and thereafter on the emotion icon all possible emotions will be displayed around the emotion icon. By selecting one of the emotions all characters in the environment will adopt the emotion.

Triggering speech action:
As a user, I want ot be able to trigger a person to talk.
When clicking on the main icon of a person, an option for speech will be displayed around the icon. Clicking on the speech option will list one/or more of the predefined words/sentences that can spoken. Clicking on one of these predefined options will trigger them.

Changing object state
As a user, I want to able to change the state of objects in the environment, so that I can influence the patient's experience.
When clicking on an interactable object icon in the environment (e.g. a TV), the options for this object pop up around the icon. Tapping these options will change the state of the object (e.g. the TV will be turned on or speakers will give an announcement).

Adding person to environment
As a user, I want to be able to add a certain character to the environment, so that I can influence the patient's experience.
The 2D map will have a sidebar (on the left of the map) containing a "new character" icon. Tapping on this will show a list of possible characters. After selecting the character by tapping it, it can be placed in the environment by tapping the target location.

Focus on patient:
As a user, I want to be able to focus on the patient throughout the procedure and whenever I move the map be able to quickly center the map back on the patient.
The user can toggle the "center map around patient" feature that automatically adjusts the view of the map according to the patient's position in order to maintain the focus on the patient automatically. Whenever the map's view is shifted somewhere else, re-toggling the "center" feature will focus the view on the patient again.

View subset of map:
As a user, I want the map to be zoomable and movable, hence be able to zoom into and shift the focus on different aspects of the map.
The user can zoom into and "drag" the map into different directions by using keyboard shortcuts in order to view subsets of the map and shift the focus on different areas.

View logs:
As a user I want to be able to view the logs of the entire therapy session, in order to recap on certain events and/or establish an overview of what exactly took place during the procedure.
The user can access the log file of the current session by using the sidebar shortcut for the log file.

User lost:
As a user I want to be able to consult documentation on the application to figure out how something works.
The user will be provided with a manual in the control menu to easily access documentation for help when lost.