I have worked in Agile development environments for the past few years and do believe in the practices applied such as scrum and being sprint focused. I have seen amazing new ideas come out of doing sprints as well as solutions to big problems that improved projects greatly. Sprints have given me the opportunity in the past to explore new options and grow my skills and creativity by rapidly prototyping new ideas.

Reading up on agile software development I discovered many methods and ways of applying agile in my correct development. I found two practices I am going to take forward into my upcoming projects to help me achieve my outcomes and goals in both design and development.

In Juhani Warsta, Pekka Abrahamsson, Outi Salo and Jussi Ronkainen peer-reviewed academic literature paper on Agile Software Development Methods I stumbled upon an agile practise I found very interesting that would help me think about design more efficiently. In section 3.7 Adaptive Software Development they describe different Agile methods of which I focussed on the section 3.9.2 Pragmatic Programming method.

From an agile viewpoint, the Pragmatic Programming methods focus is on incremental, iterative development, rigorous testing, and user-centered design. The approach taken here has a very practical standpoint as they are illustrated by positive and negative examples accompanied by questions and code snippet, where focus is put on withstanding changes that can happen within the industry or development process by designing software with a focus on change.

Pragmatic programming does not have processes, phases, distinct roles or work products. The philosophy is defined in six points, which could be stated roughly as:

* Take responsibility for what you do and think solutions instead of excuses.
* Don’t put up with bad design or coding. Fix bugs and issues as you see them or plan them to be fixed very soon.
* Take an active role in introducing change where you feel it is necessary.
* Make applications that satisfies your customer/creative goals but know when to stop.
* Constantly broaden your knowledge.
* Always improve your communication skills.

Peer-reviewed academic literature paper full reference:

Juhani Warsta, Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, J., 2002. Agile Software Development Methods. Agile Software Development Methods: Review and Analysis, [Online]. 1, 72, 88. Available at: https://pdfs.semanticscholar.org/048b/c875df549a2d1fda6d34e98fab5f72121028.pdf [Accessed 29 June 2019].

My second source is a peer-reviewed academic literature case study by William Herman Morkel Theunissen on Agile software development where I focused on section 2.4.1 Extreme programming (XP).

XP is an agile methodology that has the software team comply with well-defined and well circumscribed processes when developing software. These values need to be met before their development methodology is classifiable as XP. The most important of these processes are the four-core values, principles and characteristics XP is based on. The XP agile methodology started developing due to problems faced within software development in the form of applications not meeting the customer expectations, reduced applicability of delivered software, the inability to meet deadlines and schedules.

XP may be regarded as the most published agile methodology It is based on a series of values and practices that are applied to design and quick iterations of sprints.

The four core values of XP are:

* Communication: This involves keeping everyone on the project informed on everything regarding the project.
* Simplicity: To always make use of the simplest solution that can work.
* Feedback: Decrease iteration time between the response of doing something and getting the result. Examples of this are the time between asking a question and getting an answer, the time spent on implementing new features and confirming its success using well defined automated test cases.
* Courage: This refers to one’s ability to make those difficult decisions that need to be made and correct the project’s direction if needed.

The following practices characterise XP:

* The planning game: Always plan the scope and objectives for the next release.
* Small releases: Release the application iteratively in short cycles as updates.
* Metaphor: Use simple stories to act as guidelines for the project team on how the new feature functions and how to implement it correctly.
* Simple design: Always make sure the application is as simple as possible at any given moment of development.
* Testing: Continuous testing of code should take place. This could be unit and feature test cases that are created by the client in conjunction with developers, to ensure that the application always works as intended.
* Refactoring: Continuously simplify and restructure the system without changing its behaviour. The purpose is to improve the quality of code.
* Pair programming: Develop code using two programmers at one machine.
* Collective ownership: Code always belongs to everyone on the team and can be changed by anyone as needed. This does not mean they got to push the code to the latest version but provides an environment for growth and learning to take place.
* Continuous integration: Always integrate features as it is completed into the application.
* 40-hour week: Project team members working hours are limited to only 40 hours a week. It is believed that this increases productivity and moral among developers.
* On-site customer: A user or client who can answer and address questions as they arise from design or development should be made a part of the team when possible.
* Coding standards: Code is written according to a selected standard to ensure uniformness and optimal communication.

Peer-reviewed academic literature paper full reference:

Herman Morkel Theunissen, W., 2003. A case-study based assessment of Agile software development. A case-study based assessment of Agile software development, [Online]. 2.4, 22. Available at: https://pdfs.semanticscholar.org/b251/380a292baa020608613a2f1e8d5e293c2de5.pdf [Accessed 29 June 2019].

Reflecting upon the App jam of week three I noticed that I did apply some of the practices involved in Agile development. I found the practice of Extreme programming (XP) as I applied the planning game and simple design characteristics during my design phase and refactoring continuous integration along with coding standards in my development phase of the app jam. As issues in the application arose, I found myself using Pragmatic programming methods as thinking solutions instead of excuses, Fix bugs and issues as you see them, introducing change where you feel it is necessary to help me face the barriers ahead and overcome them.

During the app jam I did practise Agile methods but didn’t always focus on all the characteristics equally and didn’t see how it all comes together. In my continuation of the app jam project across the study block I will apply Extreme programming core values when thinking about design and creating new features for my applications as it grows in complexity.

Applying Simplicity together with the Feedback to make sure I keep designs simple and achievable will help me keep every goal within scope and help me keep to my time allocated as well as decrease the response of doing something and getting the result. Applying these practices will need to include Courage as I will need to make difficult decisions on project direction and weather doing a feature is worth it. Continues iterations also is a characteristic I will apply as I push code up and update the application as new features are completed. Refactoring code as I go along and not lingering on this characteristic too long as it was what took up most of my time during the app jam although the features worked.

Pragmatic programming six points philosophy will be of value when I am in the developing phase of the project as problems and barriers can arise that require quick thinking. By having a “think solutions instead of excuses” mind set will help me see through the barriers and seek solutions by Constantly broaden my knowledge and finding solutions to the problems that will arise. I will also take a more active role in introducing change where I feel it is necessary using the new knowledge gained from finding solutions and broadening my knowledge of the issue of new feature.

When I see an issue, I will need to either fix it immediately or log the bug on Trello to be fixed sooner then later. Not putting up with bad design and coding will need to be implemented as a standard to avoid a pile up of issues and fixes breaking design going forward.

I will also need to know when to stop. Making applications that satisfies my creative goals but is also within my scope is the key. Knowing when to stop combined with Extreme programming core value Simplicity will keep my creative vision and scope aligned to help produce applications that meet my goals and is build as intended from design. Even if ideas end up failing its not failing at all using Agile software practices. The questions asked and the answers found builds a successful application going forward.

My progress so far with my AR application has been slow due to the app jam. I learned so much with the Week 3 App jam and especially combining it with this week’s Agile practice. It highlighted how I could improve my approach and make sure I don’t fall in the same traps as before.

The current status is that I have completed my first objectives set out that included getting the application building for android using the unreal engine and making the AR recognize images and flat surfaces. I have also built my two fish and have them rigged and a small amount of animation applied. I have also built a fish tank and applied glass and water shaders that has been tested on mobile platform.

I have lowered the scope of the project due to what I have learned from the app jam and this week’s agile methodologies and have decided to apply SMART principles to solve the small goals I have set out for myself. I have decided to go from two fish to one and make sure all features work on it before even looking at a second. My next goal is to implement the GUI and be able to feed the fish and adjust the health status that is displayed to the user.

The specific goal set out with this feature is to have the fish recognize there is food in the tank and to go eat it. This is accomplished by having the fish know where the target is located by using path finding and destroying the object once consumed by the fish triggering +1 on hunger displayed on the GUI.

I can achieve this goal by applying simple a\* pathfinding, making the food the objective for the fish. A\* pathfinding solutions are widely available like the one provided by Monokkel, M. 2014. My A\* Pathfinding blueprint or: How Not to Build a Blueprint. 8th August. Unreal Engine Forums. [Online]. [27 June 2019]. Available from: https://forums.unrealengine.com/community/community-content-tools-and-tutorials/12979-my-a-pathfinding-blueprint-or-how-not-to-build-a-blueprint.

The food itself will be physics driven as it falls to the floor of the tank, but I will leave them in place for now as my main focus is to get the pathfinding going. I will make all this possible by unitizing Unreal engine blueprints for rapid prototyping. This will be a week sprint for me to achieve this goal and I feel like it falls within my skill level as I have done things similar before.

Having this feature completed would help my AR fish tank simulation closer to being like the real thing as fish need to eat right. Completing this goal will help me start the AI phase of the application and plan for future integrations that need to take place around behaviour. It will lay a ground base for me to work off going forward and a start to my proof of concept.

Giving myself a week to implement the feature is reasonable. I will be implementing the blueprint to my character in three days and spend another day or two building out to mobile and testing. I also want to leave me some time to refract code if needed and make sure everything upholds to standards and I will be pushing this up at the end of the week and applying it to my project. I left one day apart for me to reflect on the feature and how I can do things better going forward.