Design Report – CS Open Days

University of Twente

Samuel Coste – s2245973
Hans Goor – s2311720
Tim Koree – s2341182
Ronan Oostveen – s2285371
Reinout Vos – s2385058

Contents

Abstract

Tim Koree

The CS: Open Days project explores the use of gamification to intuitively teach programmatic thinking. The application aims to guide, entice, and teach prospective students about creating their own computationally intensive solutions and help them interact with Teaching Assistants during the open days. Its UX- and UI interaction are focused on occupying the user for half an hour, giving them the opportunity to learn about the University of Twente through a fun and engaging workshop. Using technologies such as NextJS, Typescript, Tailwind CSS and GSAP, the application creates a strong foundation for future developers to build upon.

Planning and Structure

Tim Koree & Reinout Vos

INTRODUCTION

The team put a considerable amount of time into planning. While it is impossible to foresee every unexpected obstacle, it is possible to plan considerable room for unexpected circumstances. This entails primarily, to start early and avoid unnecessary crunching at the end of the project cycle. This includes starting with writing a report early, having a functional minimal viable product and planning in time to test the quality of the product.

SPRINTS

To ensure a consistent work-schedule, the 10-week project cycle is divided into 2-week sprints. Making use of tools such as Discord, WhatsApp, Google Calendar and more, the team aims to effectively communicate deadlines, schedule meetings, discuss important topics and additional issues. At the start of each sprint, the primary objective will be set with a task division. To support the effectiveness of sprints, daily stand-up meetings are organized. These function as a method to inform and evaluate other members of progress and issues. To ensure peer review session were sufficiently prepared for, the team internally discussed what presentation format would be most effective. The two chosen presenters would create presentation slides that encapsulate the progress made within the last print iteration. Additionally, when planning for reports like the project proposal, design report or reflection component essay, the team's focus remains to start as early as possible when the planning allows. For each of these mentioned assignments, a task division is created. The work is then reviewed by all members to ensure its academic quality.

ADDITIONAL MEETINGS

As the application is produced with intent to deliver to a customer, it is of high importance to collaborate effectively with the client. To ensure this, bi-weekly meetings are planned in at the end of every sprint. This way the progress can be effectively displayed and additional feedback can be implemented for a next iteration. Additionally, it provides time for questions and clarity on the requirements and possibilities of the project. To communicate with the project supervisor, meetings are planned when bigger progress is made within the application and feedback is desired. This includes feedback on the academic quality and additional requirements.

TASK DIVISION

To ensure an even task division, the project is divided into three sections. First, the game logic. This entails designing levels as well as handling all the input the player gives. For example, moving the block around the maze while making sure it cannot phase through walls. This will serve as a sort of back-end for our application. This is handled by members Hans and Samuel. Second, the code builder, loader and design. This includes the functionality to create programs through moving pieces of code, creating a debug-mode, allowing for a full-screen view of the program and creating the overall branding and design of the application. This is handled by Tim. Due to his experience working professionally on web applications, he works alone and aids others whenever necessary. Third, the maze, star-system, level-switcher and onboarding. This includes displaying the game-logic in a visually appealing manner, allowing for switching levels, giving users performance feedback through the form of a star-based rating system and creating an onboarding to explain functionalities of the application. This is handled by Ronan and Reinout.

Design trajectory

Reinout Vos

SPRINT 0

Prior to meeting the client, we wanted to already have a design to present to them, so we quickly decided on a design and branding. As we are making a workshop for the university, we wanted to include the university's styling by making use of the same colors as well the simple geometric shapes. This resulted in a clean a simple, yet clean design for our application. After creating an aesthetic for our product we brainstormed for a bit to think about the actual workshop itself. We ended up being heavily inspired by a programming language called Scratch which is designed to help visualize programming concepts by allowing the user to drag and drop code blocks to create a program. This idea fits our project very well as our goal of introducing the workshop attendees to some basic programmatic thinking is remarkably similar. In order to gamify this process of teaching programmatic thinking we decided to use the code block system to try to solve increasingly more difficult puzzles in the form of mazes. Using this maze design, we have a lot of freedom to expand once the simple maze gets repetitive. For example, we can add enemies the player has to avoid, moving walls, etc.

SPRINT 1

During the first meeting with the client, we presented the above-mentioned design and workshop idea we had come up with and they were enthusiastic. However, we did receive some useful feedback about desired ideas and requirements: The biggest point is that the game should get increasingly more difficult up to a point where the levels become almost unsolvable. The idea behind this is that the players should be motivated to talk to peers and teaching assistants present at the workshop. This includes asking questions, discussing their results, etc. Furthermore, we discussed some smaller features, like the idea of a debug mode and onboarding. Just like in any ordinary programming language this feature can be used to help the player once it gets stuck on a level by allowing them to go back and forth through the program line by line. The onboarding is used to help players with limited programming experience. Once a new form of logic is introduced, for example an if-statement, we will give the player an option to get some extra information as to how said if-statement works.

SPRINT 2

As the general ideas and requirements were discussed in the previous meeting, this meeting was a bit more technical. The biggest discussion point was how we should handle the nesting of statements. We felt allowing for nesting would be a great addition to the workshop, however it would be difficult to implement. Eventually, the client said this feature was too essential to leave out, so eventually we decided to give it a go. This warranted a change in our design, since the current code section is quite narrow, so nested statements would eventually become unreadable once you nest them deep enough. In order to solve this, we designed a full screen mode. Now the code section can be expanded to full screen while a smaller version of the level will still be displayed to prevent having to switch between the two views all the time.

SPRINT 3

Between the second and third meeting with the client we had the first meeting with our supervisor. They gave us some useful ideas we wanted to discuss with the client as well:
The first idea was to give the player all the code blocks needed to solve a level, leaving the player to place them in the correct order. A way to make this more challenging would be to also throw in some 'fake' code blocks in there which the user should avoid.

SPRINT 4

As this was the last meeting during our development phase, we discussed what features were could still implement within these last two weeks. During this sprint, before meeting the client, we had the chance to do some user testing during one of the UT Open Days. During this test we found out that the application still contained quite some bugs, so during this meeting we mentioned this to the client. Furthermore, because we still had to fix these bugs as well as finish some remaining essential features for playing the game, such as making a score system or being able to swap between levels, implementing features like the beforementioned fake code blocks, unfortunately did not fit within the scope.

SPRINT 5

This sprint discussion included little feedback or comments. Its focus was primarily on displaying the progress made within the development of the application. We mentioned the small issues we still had to fix before the deadline and the client seemed okay with it.

DELIVERY

A final meeting was planned to hand over the application. We showed the client the last features we implemented and discussed how the code should be handed over. Concluding the meeting by explaining the structure of the code and setup of the application to the client.

Design and Academic justification

Tim Koree

OVERVIEW

To understand the application's final implementations, it is relevant to first analyze the client's original requirements. The client communicated their high value on requiring programmatic thinking and for its function as a communicative tool. The UT open days serve to entice prospective students in joining a program at the University of Twente and the application is aimed to reflect this. Its aim is to occupy visitors for a duration of 30 minutes and allow for a fun and engaging experience. This includes, functioning to display what programmatic thinking entails and allowing visitors to easily communicate with Teaching Assistant's through questions and feedback. The application's primary objective becomes to bring prospective students a fun, informative and memorable experience. Simultaneously, these characteristics separate this project from existing gamifications of programming. Rather than solely being a learning or gaming platform, its focus lies on connecting prospective students with the University of Twente.

GAMIFICATION

To ensure the project's primary objective, the application must make the complexity of programming clear, but accessible enough to motivate beginner programmers to participate. Research by the Bournemouth University displays that gamification can be used to "increase performance and engagement of students" (A. Sahri, M. Hosseini, K. Phalp, J. Taylor, R. Alie, November 2014). Among other methods, gamification can motivate users by rewarding users through performance feedback or other rewards. Within this project, this concept is applied by transforming complex problem-solving into a fun and interactive game. It helps users understand what programming entails in a very fundamental sense but can also display the satisfaction and enjoyment that complex problem-solving can bring. This application gives users performance feedback to help gauge the relevance of efficiency and performance within programmatic thinking. To aid in communicating these functions, the addition of an onboarding and familiar design patterns are used. The use of a 2D-grid and the applications movement all adhere to these familiarized standards, popularized by games such as `Pac-Man` and other maze solvers. The use of gamification helps lower the barrier of entry for prospective students and aids to create a memorable and engaging experience.

LEVEL DESIGN

The motivation and engagement are also linked to the quality of the experience. The `CS: Open Days` project must ensure users cannot finish before the total duration of the workshop, additionally it aims to provide users with challenging and informative levels to increase the quality of the experience. First, to ensure users cannot finish within the time-limit, the game houses a total 49 levels that increase in difficulty, additionally the application slows down users using animations and motion. The quality of the experience is strongly linked to quality of the product, in which the intent of each level becomes key for the quality of the level design. Levels are focused on introducing and applying increasingly complex programming instructions. This is done using a whitelist, onboarding, level-design and the addition of a scoring-system. The whitelist restricts access to instructions during earlier levels, new instructions are introduced with an explanation within the onboarding and practical application of the instruction. To encourage the use of more complex and efficient instructions, the addition of a scoring system is put in place. The star-rating system within the `CS: Open Days` application aims to increase this curiosity and engagement. However, the level design extends beyond the quality of the experience within the application. The increased difficulty of levels additionally increases the complexity of solutions. The addition of a `debug-mode` functions as a tool for prospective students to step-by-step iterate over their code. However, it also serves as a method for Teaching Assistants to connect with prospective students. Users may find themselves stuck and are encouraged to ask for help. A teaching assistant can use the debug-mode to aid in solving the level and simultaneously inform prospective students about the University of Twente.

LANGUAGE DESIGN

Operability and intuitiveness of the `CS: Open Days` application is essential to avoid frustration and decreased motivation amongst prospective students. The function of the application is to require programmatic thinking and encourage users to create computationally complex solutions. Due to the characteristics of the game, it is relevant to analyze its implications from a language design perspective as the tools used within the application allow users to communicate instructions through a written language. Its requirements primarily include that it must be clear how these instructions relate to the game-logic. A user must be able to easily read and comprehend the structure of the language and written instructions. To structure this language, the idea of Functional Programming was adhered to. For further understanding, the structure of popular programming languages such as Python, C and Java was analyzed and deconstructed, as was work by researchers such as P. Hudak and his paper on the "Conception, Evolution and Application of Functional Programming Languages." (P. Hudak, September 1989). Using these sources, a very simple functional language was constructed, in which users can communicate forms of navigations through a 2D grid.

BRANDING & MOTION

The design choices within the `CS: Open Days` application can all be justified from the perspective of the `rationale`. The team's philosophy on branding is the concept of consistency. Therefore, every design choice is made from the perspective of the branding and motion language. The concept behind the branding stems from aiming for a non-intimidating look, versatility and connecting the `CS: Open Days` project with the University of Twente without adding its logo or name. To achieve a non-intimidating aesthetic, the use of simple chapes and rounded corners is used. The colour-scheme reflects the UT's branding and creates great contrast between the primary, secondary and tertiary colours. The versatility and high expandability are achieved through the consistent design choices and colours throughout the branding. It creates great potential for adding additional features that can seamlessly fit within the design of the application.

TOOLING

The application’s technologies have a primary focus on maintainability. The `CS: Open Days` project is built with NextJS and TypeScript in its base. NextJS extends React with further optimization and server-side rendering, the framework is an excellent choice due to React’s popularity and TypeScript’s readability and clarity. To create complex, but beautiful animations, the team used GSAP. Its functionality to easily created layered animations with beautiful easing’s makes it an obvious choice for speed and readability. The use of Tailwind CSS and SCSS further speeds up workflow for efficient coding and the use of Studio-freight’s Lenis scroll creates a beautiful smooth-scrolling experience whilst retaining the browsers native behaviour. To create the designs and icons, Figma is another industry standard that the team relied on.

Quality control

Samuel Coste & Hans Goor

GOALS

We wanted the ability to show our client the progress during the regularly scheduled meetings, without having bugs or other faults on display. Besides this, we wanted a nice development environment where we could show our application and not mess up critical API methods during heavy development. To achieve this, we needed to implement a continuous integration and deployment system with unit tests. We opted for automatic unit testing with manual system and integration testing. This is because we work with several different components that are split into separate code repositories and writing automated tests combining them would not be time efficient.

CI/CD

For our main development platform, we chose a self-hosted instance of GitLab. This software not only has the feature set for hosting a git repository, but for issue management and continuous integration / deployment too. Since we did not want to set up additional software and because we were all familiar with GitLab, we opted to use these integrated features. In the first sprint we set up our base repositories and immediately set up continuous integration for our logic repository and continuous deployment for our frontend repository. Both projects use NPM for dependency management and running, so this was easily set up.

WORKFLOW

Our workflow in GitLab consisted of branching from a stable development branch to create a feature and merging back into the stable branch upon completion. Because we were strict with internal coding guidelines, code would not be merged without tests into the logic repository. This allowed us to incrementally add tests and assert the functionality of our code. Upon merging, GitLab runs our continuous integration pipeline checking if all tests pass with the newly added feature. Should any one of these tests fail, then the code would not be merged. We also always had another pair of eyes review the new code before merging.

DEPLOYMENT

Besides our automated unit testing, we also had continuous deployment for easy showcase of the latest development and production versions of the app. We set this up in sprint 1 for our front-end using a dedicated server and 2 domain endpoints. One hosting the development version, the other the production version. Every time a push would be made to either the development or main branch, the pipeline would run. This pipeline consists of linting the code, building the release version of the app and creating a docker image to run it. Then, it would reach the deployment stage where it would stop and run the docker image on the dedicated machine. Ultimately, this means within 5 minutes of a merge being accepted, the app would be live for display.

NOTES

We opted to not write unit tests for the frontend. Instead, we did regular manual checking and verification of features. The time gained with automated frontend testing would not surpass the initial time investment to set it up. Moreover, we already had unit tests for the logic behind the front-end. Therefore, we knew functionality was correct before integrating it into the front-end.

Conclusion

Ronan Oostveen

The careful consideration done during the set-up and design phase of this project significantly helped us later in the project. First, the decision on the specific technologies allowed us to quickly design, develop and troubleshoot our application. The planning and especially the daily stand-ups also aided us in making serious progress early on. The task division also prevented a lot of confusion. And in the bi-weekly meetings with the client and supervisors, we got useful feedback which helped us plan the next sprints.

To conclude the design justifications, we were able to achieve all the requirements. The application has an accessible interface but can keep an experienced user busy for 30 minutes. The design and branding are consistent throughout the application and consistent with the university branding.

Overall, we are very satisfied with the progress made during this project and proud of the final product. The user testing showed us that the product achieves the desired goals of the client. In this paper, we also explored the academic background behind the design of our application and used this to support our design choices.

References

A. Shahri, M. Hosseini, K. Phalp, J. Taylor, R. Alie. (November 2014). Towards a Code of Ethics for Gamification at Enterprise. Retrieved from https://www.researchgate.net/publication/275968573_Towards_a_Code_of_Ethics_for_Gamification_at_Enterprise

P. Hudak. (September 1989). Conception, Evolution and Application of Functional Programming Languages. Retrieved from

http://www.dbnet.ece.ntua.gr/~adamo/languages/books/p359-hudak.pdf