Project title: ‘Last Hope’.

Team AC: Daniil Gannota, German Mikulski, Aksel Cakmak, Gabriel Vanca.

Introduction

The main aim of our project is to teach and inspire a complete beginner to learn basic programming skills. Therefore, there are three main areas we concentrated on: simplicity of the material, keeping the user entertained and independence from any particular language (the teaching materials should visually illustrate abstract notations and rules with no accent upon a single language such as C++ or Java).

Our team has decided to make those three points to be our main as due to the fact that this is the way we would like to had been taught initially. In our view, this approach will encourage further interest in programming or at least opens the basics of computing to beginners; thus, help them learning any programming language later.

The way our project follows the requirements and corresponds to what is stated on Moodle is by having a Unity program, a game. The game that can be ran on any computer, named 'Last Hope', is a visualiser of such abstract expressions as “if” or “while” loops, etc. The application tries to educate a person, who is completely new to programming, those basic logic statements showing the result displayed in a game: it has a character – that is affected by the commands – and a field/board full of obstacles to be overcame. The user controls a character in a map/maze, with the help of basic programming commands, that put together, form a logical series of steps that the character is going to execute.

As the user completes levels, they become more and more complex. At the same time, we introduce other logical statement, for example, “do while”, “if else” and “for” loop. We intend to make users think and develop their understanding of the concept behind them by increasing the number of steps that have to be predicted in order to complete new levels.

Furthermore, the application is intended to inspire young minds with computing. While playing the game and observing the way programming affects the behaviour of the character, they might start wondering of other ways that programming can be used as well as research the way our application is made.

MoSCoW and Team Management

In order to be as efficient as possible, we first established features that ‘must be’, ‘should be’ or ‘could be’ as well as ‘will not be’ included in the project to prioritise the objectives of the development game.

General description: a game, where movements of a player are controlled by dragging and dropping specific blocks, representing pieces of code, e.g. "go_forward" function that will make the character move forward. Target age-range: 10-15 years.

Must:

- A map split into cells (fields)
- Some cells are inaccessible (obstacles)
- Compose code from drag and drop blocks
- Characters behaviour determined by the code blocks’ content
- Press "Run" button to run code
- Combine "go_forward", "turn_right/left", "if", "while" blocks
- Complete levels by reaching certain point

Green for completed tasks. See more in appendix.

Should:

- Add "if else", "do while", “for” statements
- Have more sophisticated graphics
- Enable user to save game progress
- Display C code and some explanation of it
- Hint/model answer if user is stuck
- "Jump", "Attack" functions

Could:

- Add "compile" feature
- Have tutorials – description of each statement in real programming
- Support different languages: Java, C#…

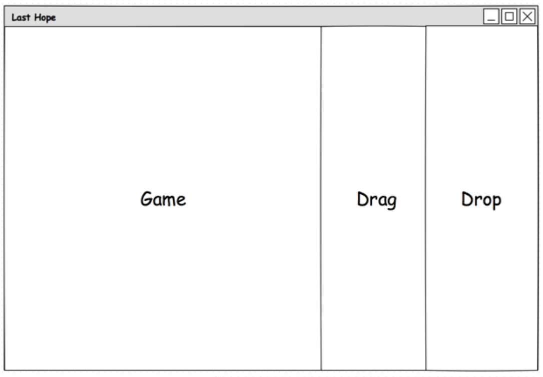
- Add several points to reach - rating of how well performed
- Several textures for the environment/player
- When the user collides with certain obstacles, animation occurs
- Ability to see ‘Model Answer’ – easiest way to complete each level

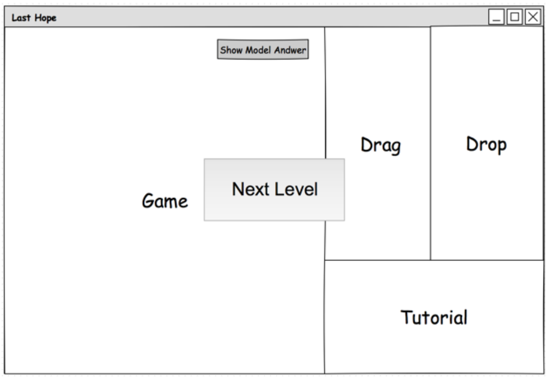
Won't:

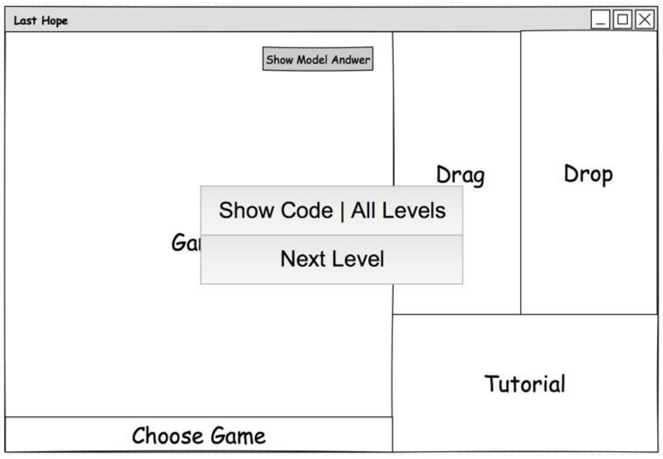
- Be too complex in terms of programming
- Be based on one programming language

Note: ‘Must’ section is the whole work planned. ‘Should’ and ‘Could’ lists show additional features that we considered, but they may not be implemented due to timeframe.

Furthermore, we created user interfaces that reflect each major iteration of project development. These design concepts visualise MoSCoW priority list, and therefore help managing tasks between team members, so that everyone sticks to what is of a higher priority. Note that those steps have been changed throughout the project and only the final version is attached. Steps 1, 2 and 3, respectively. Pictures are a reflection of MoSCoW, description is above.







Last but not least, we conducted our work through the use of a task manager (Todoist), where each person is assigned a certain piece of work and everyone else is able to track team progress. This and other software that provides multiple users sharing capabilities of files and code, such as Google Drive or GitHub, made our collaboration much more efficient.

Current State and Future Steps

Currently, we have completed the most difficult part of our work. That is creating the base for all game levels:, 'drag/drop' functions, character behaviour as well as primary obstacles and decorations. This allows us to conveniently use those basic features in the development of new levels, which will only require a creation of new maps.

Our next steps will be to add more features to the user interface as well as more teaching materials that would describe each piece of code showed, for instance, "if else" statements in C# or Java. Therefore, we intend to not only show the abstract concept of logical expressions, but also relate to some programming languages. This is not to say the application will be single computing language based, we are enthusiastic to implement a lot more examples. Thus, by pushing 'Show Code' button the user will be able to select preferable language(s) (e.g. Java or C) and see the actual code in it.

Finally, we are inspired to continue the development of the project even after the final submission. There are a few features that may take a lot more time to be implemented. However, despite our huge plans, good planning sets the course of the work so that we primarily deliver finished product, and only then add new features.