PROPOSAL: A Platform Game in Elm

Yang, Jessica jessicayzt@alumni.ubc.ca
van der Kooi, Tim vanderkooi11@gmail.com
Hong, Karen khong@alumni.ubc.ca
Rodgers, Laura laurarodgers@alumni.ubc.ca

OVERVIEW

We plan to build a platform game in Elm, a somewhat unusual programming language none of us have studied previously. Elm is a functional language specialized for building robust web applications. This proposal introduces Elm and its features, how we will apply these features in our project, and outlines our approach to meeting all the project milestones.

1. INTRODUCTION

Elm is a functional language based on the Haskell paradigm but designed for straightforward use in front-end web applications. It compiles to JavaScript and uses strong static typing, which has led its architects to advertise that it guarantees "no runtime exceptions." Through immutable data structures, Elm boasts increased performance by allowing for shared memory whenever possible, reducing recomputation. It allows strategic use of lazy evaluation and higher-order function applications, all of which are not just esoteric luxuries, but production-oriented practical features that can, when used skillfully, dramatically improve performance. Elm lends itself to a declarative programming paradigm, with high readability and extensibility.

2. OUR APPROACH

The 80% - Background Research Report

To complete this milestone, we plan to research extensively on the syntactical and semantic idiosyncracies of Elm (referencing [1,2]), comparing it to the more popular, less declarative web development languages (such as bare JavaScript, in which we will use [3] as a reference) our team has experience with, and investigate features that will be beneficial for us to use in the implementation of our game.

We will describe the declarative paradigm used by Elm programmers, the ins and outs of the Elm type system (outlined in [4]), and strategies to optimize performance in Elm (outlined in [5]), particularly in how lazy evaluation can be used to optimize an animation-heavy reactive application like ours. We will focus our research on areas that will be useful to us, such as Elm's Graphics.Collage package for freeform graphics and static analysis strategies encouraging one to stray from standard web development idioms. Researching this will allow us to demonstrate the benefit of structuring our game using Elm's functional component system: inputs, model, update, and view (in which we will use [1,2] as references).

It is at this milestone that we will articulate the key concepts undergirding the Elm language, and why the creators of Elm found it worthwhile to design in the first place.

The 90% - Proof-of-Concept and Plan

To achieve this milestone, we must first get comfortable programming in Elm through hands-on practice. At this step, we plan to lay out the outline for the game's design, breaking it down into specific tasks that we can fulfill.

To complete the proof-of-concept, we plan to build a prototype that demonstrates the

necessary elements of a platform game, such as user input, graphics, and reactivity. Though this prototype will not be fully featured, it will communicate that the end product we are aiming for is possible. To parse user input and render freeform graphics, we will utilize Elm's Graphics.Collage library package among others. We will also begin to make use of Elm-specific optimizations that will improve our game's speed and responsiveness. Additionally, we will be demonstrating how programming in a functional paradigm will allow for cleaner event handling, compared to the often messy use of callbacks in JavaScript.

Along with the prototype, we will present an updated plan detailing what is necessary to create a fully featured game (i.e, what the prototypical game is missing).

We plan to implement these features in our prototypical platform game:

1. Simple motion commands in a side-scrolling environment
    a. A player of our game should be able to move their character left, right, or jump, based on keyboard inputs
2. Left or right movement should result in a side-scrolling screen
3. Simple environment
    a. The character should respect the boundaries of objects in its environment, which will be made up of suspended platforms
    b. The positions of the suspended platforms should be randomized to some degree
    c. The environment should be generated "on demand," (ie. as the character proceeds to the right)
4. Simple hazards and health bar
    a. A loss of a certain number of health points after collision with certain pre-determined obstacles

The 100% - Final Project

At the final project milestone, we will be implementing our core game fully and testing it, using our prototype from the proof-of-concept milestone as a starting point.

We plan to implement these additional features in our final game:

1. Additional environmental hazards
    a. Various ill effects (ex. loss of life, loss of a certain number of health points, or slow speed) after collision with certain pre-determined obstacles, such as bottomless pits or spikes
2. Collectibles
    a. Addition of collectibles that have a positive effect (ex. increase score, increase health points, invulnerability, increase speed)

The Poster

With the poster, we plan to outline the key features of Elm, especially its static type-checking system, and why one may want to use it to build a robust and fast web application over existing and more popular languages. For the latter, we plan to outline common problems one may have in web development, such as runtime errors in JavaScript, or unneeded evaluation rendering unseen frames or structures in animated applications, and how Elm can solve these common problems. We will put more emphasis on readability than amount of information for the poster, using images, diagrams, and screenshots over text whenever possible to communicate our ideas. We also

plan to have our game completed by this time so we can demonstrate to our peers what we have created with Elm, with a laptop demo.

3. STARTING POINTS

We have gathered several key articles and online resources on specific Elm features which will aid in the development of our project:

On Elm Semantics

[1] Czaplicki, Evan. “Introduction.” An Introduction to Elm, guide.elm-lang.org/.

[2] Czaplicki, Evan. Elm, Evan Czaplicki, 2012, elm-lang.org/.

On Comparison to JavaScript

[3] Rolo, Pedro. “Elm = Javascript Reinvented.” https://www.imaginarycloud.com/blog/elm-javascript-reinvented-1-overview/.

On the Elm Type System

[4] Waselnuk, Adam. “Understanding the Elm Type System.” Adam Waselnuk - Front End Web Developer, Adam Waselnuk, 27 May 2016, www.adamwaselnuk.com/elm/2016/05/27/understanding-the-elm-type-system.html.

On Performance Optimization Strategies

[5] Czaplicki, Evan. “Blazing Fast HTML: Round Two.” http://elm-lang.org/blog/blazing-fast-html-round-two.

We have also compiled a list of additional readings:

Articles and Online Resources

Reimann, Dennis. “Elm, Functional Front-End Development and Why You Should Care.” Changelog, Changelog, 8 Feb. 2016, changelog.com/posts/elm-functional-front-end-development-and-why-you-should-care.

Eriksson, Nils. “Move fast and don’t break things: Running a startup on Elm.” https://medium.com/the-ahead-story/move-fast-and-dont-break-things-running-a-startup-on-elm-b5491082fe8b.

Online Books

Poudel, Pawan. Beginning Elm. elmprogramming.com/.

Online Talks by Evan Czaplicki, the Creator of Elm

Czaplicki, Evan. “Let's Be Mainstream! User Focused Design in Elm.” Curry On, Curry On, 14 July 2015, www.youtube.com/watch?v=oYk8CKH7OhE.

4. SUMMARY

We are building a platform game in the form of a web application in Elm, a language that none of us have studied previously.

REFERENCES

Czaplicki, Evan. Elm, Evan Czaplicki, 2012, elm-lang.org/.

“What Is Elm?” The Pragmatic Studio, The Pragmatic Studio, 2005, pragmaticstudio.com/blog/2015/7/23/what-is-elm-qa.