Learning Games Project Requirement

Software Design and Development I

Fall 2015 CS 4560/5560

Purple Cobras

Version 1.1

Last Updated: October 13, 2015

Project Team: Emily Moore

Jordan Leach

Charles Chen

Kevin Xiao

Ryan Miers

Kameron Feola

Client: Dr. Liu

Executive Summary

The bulk of the project is creating an API to visualize data structures such as elements of an array, frames per second, or even mouse coordinates. Ease of use is also a major factor in our project as the typical users of API’s are developers themselves. Our goal is to allow developers to easily implement a visualization into their project through the use of our API.

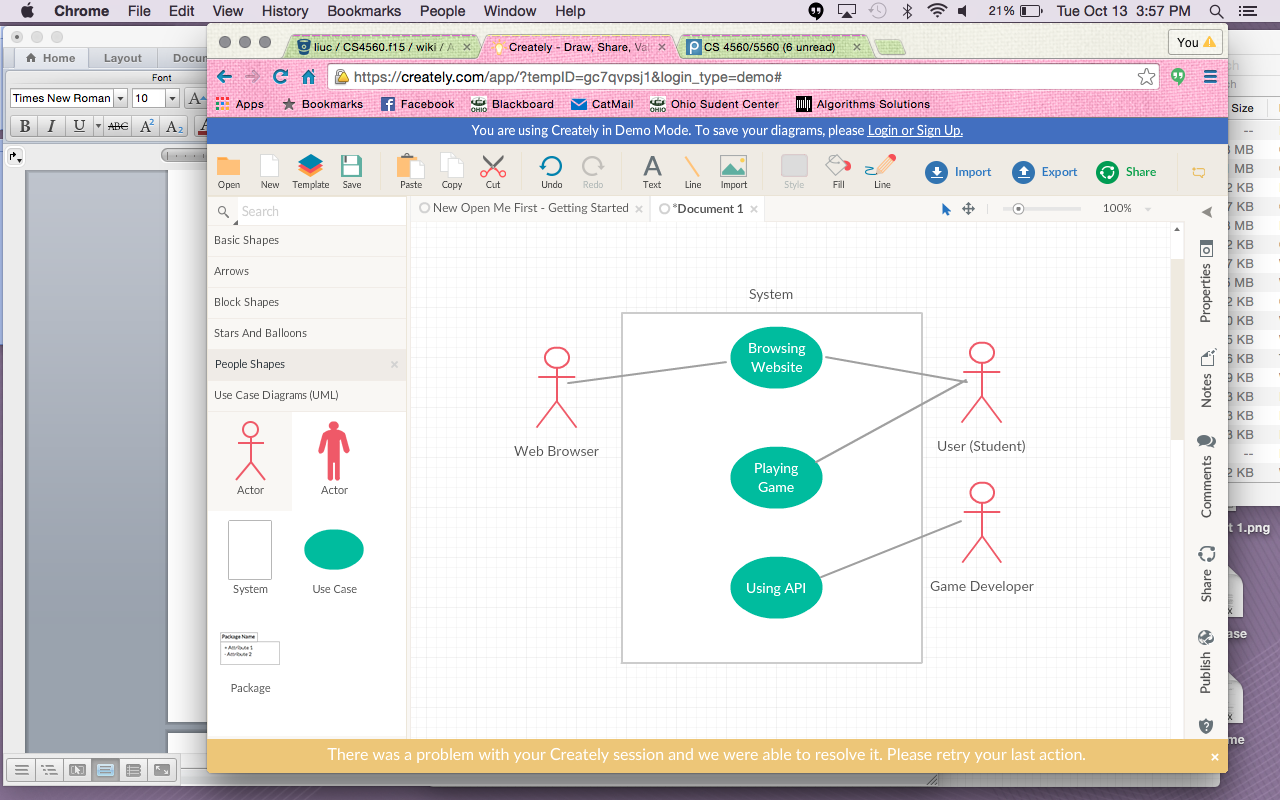
There are 5 main functions the developer will need to utilize. A call to specify what type of structure (array, stack etc.) or element the developer wishes to visualize. There is also a call to store the data that will be visualized into the API object the developer is working with. Next there is a function that allows the API to identify where the visualization will appear on the webpage. Once all the information has been loaded into the API; a function called "visualize" can be called to fully visualize the data. This function only visualizes the original data so another call must be made to the function "update" which will update the visualization, if the data has changed.

In order to test our API, we will create a simple learning game. This game will be a game centered on Boolean logic. The player will control a logical input. This input will either be “true” or toggled to “false”. The state of the player could be shown with our API whether they are true or in false. There will be a series of logic gates with given inputs coming from the top of the screen. These gates will have only one open input that the player must satisfy. For example if they player is in “true” and the gate the player wishes to collect is an OR gate with one input that is false, the player will obtain that gate. If the player does not satisfy the gate, for example, a player in “true” mode tries to collect a NAND gate with an input “true”, the player will lose a life. Lives will be kept within an array with a maximum of 3 lives. Future games could be created for testing purposes.

Overview of Requirement

The purpose of this project is to create a website that will help students grasp educational concepts through the playing of a game. The targeted audience consists of middle school students as well as high school students. Alongside the creation of a couple educational games, we are tasked with the responsibility of creating a visualization API. This API should be able to obtain certain data given by the game developer and visualize that data to the screen. This visualization could be an array, stack, a sorting algorithm or even frames per second. In general the API should be able to work with any learning game (assuming the API holds the ability to visualize what the game developer wants).

Individual Use Cases



|  |  |
| --- | --- |
| Use Case Name | Using API |
| Participating Actors | * Game Developer |
| Entry Condition | The game developer successfully downloads/includes the API in their project |
| Flow of Events | 1. The developer instantiates an API object 2. The developer specifies the elements within their game that should be sent to the API system 3. The visualization data is returned from the API system and displayed according to developer’s parameters |
| Exit Condition | 1. The API successfully visualizes the given data 2. The developer decides to not use the API |
| Exceptions | * The API fails to visualize the data * The developer incorrectly uses the API   + Syntactically, semantically, etc. * The developer incorrectly includes the API * The download of the API fails |
| Special Requirements | * The developer is given extensive documentation (including examples) of how to properly use the API |

|  |  |
| --- | --- |
| Use Case Name | Browsing Website |
| Participating Actors | * User * Web browser |
| Entry Condition | The GUI of the website is successfully loaded onto the users web browser |
| Flow of Events | 1. The user enters the URL belonging to the Learning Games website 2. The web browser sends a request to the specified server 3. The webpage is returned and loaded into the browser 4. The user is met with multiple options at the Home page:    1. View the “About” page which explains the purpose of the website    2. View the selection of available games to play 5. Upon choosing, the web browser sends a request for the new page 6. The new page is loaded onto the web browser for the user to view |
| Exit Condition | 1. The user closes the web browser 2. The web browser fails to load one of the options selected by the user 3. The server crashes |
| Exceptions | * The user enters the wrong URL * The server does not send the correct information to the web browser |
| Special Requirements |  |

|  |  |
| --- | --- |
| Use Case Name | Playing Game |
| Participating Actors | * User |
| Entry Condition | The user successfully navigates the webpage and selects a game to play |
| Flow of events | 1. The user selects the game of their choice 2. A display window is created during gameplay 3. Visualization elements are displayed in the window alongside game-play 4. The user continues playing the game    1. Visualization is modified based on the current state of the game |
| Exit Condition | * The user closes the browser * The user leaves the website |
| Exceptions | * The API fails to visualize the data |
| Special Requirements | * A brief description of how to play the game is presented to the user |

Minimum Viable Product

The Minimum Viable Product (MVP) that we will produce will include a Learning Game and a Visualization API. The Visualization API we develop will be implemented in the learning game, and will have the capability to visualize elements of the game in real-time. The API’s functionality will be easy to use and easily implemented through well-structured and well-documented code.

Technology Platforms

* D3.js
  + <http://d3js.org/>
  + D3.js is a JavaScript library that assists in creating visualizations from sets of data. Our reason for including this is to make more detailed visualizations and to possibly minimize and simplify the amount of code we write. We believe this will be used more for the visualization API although some features of D3.js may also be used to implement our game.
* D3plus
  + <http://d3plus.org/>
  + D3plus is a JavaScript library that extends the functionality of d3.js. The reason for inclusion is to possibly add more functionality to the D3 code we write. This library also claims to simplify the code written for d3.js visualizations.
* Processing.js
  + Processingjs.org
  + Processing.js is a JavaScript port of the processing visual programming language. The reason for including this library is to offer an alternative for visualization alongside d3.js. We believe that certain aspects of our game may be easier to visualize with d3.js and some may be easier to visualize with processing, therefore we have included both.
* Yui Library
  + <http://yuilibrary.com/>
  + YUI is an open source JavaScript library for crating interactive web pages. We believe that this library can assist us in making our game and making our visualizations for the API.