Dirt Bags Final Report

CS4640

12/7/19

Jonathon Smith & Giovanni Varuola & Hank Gansert

Abstract:

For our project, we created a side scroller dirt bike racing game. The game functionality is limited due to time constraints that we experienced. The only functionality currently is moving the bike forward with the right arrow key, and applying a speed boost with the spacebar key.

There is login and registration functionality, so that the user's profile and purchases can be stored in the database. We wanted to have a way for users to be unique and have their game data saved for future use, so a user can come back to our site and still have their times and unlocks. We also implemented a game store, where the user can buy speed boosts and skins for their bike.The game store is connected to the database, and the users purchases are persistent. We wanted users to have a way to get better times even if they are not the best player and stylize their racers to make them feel unique. There is also a leaderboard page to display the user's scores on levels. We wanted the users to compare their scores with other players, so they can see how good they are at the game and find ways to improve themselves.

URL:

Jonathon Smith: http://3.90.148.153/

Giovanni Varuola: http://54.211.232.219/

Hank Gansert: http://54.225.18.80/

Introduction:

Our project is a dirt bike racing game deployed to AWS as a web application. We decided to do a game versus the learning outcome tracker because we were more interested in learning about the process of creating a game than working on the learning outcome tracker. We decided that starting with something fresh would be more enjoyable as we were all a little burned out with the learning outcome tracker.

We designed our game to be similar to Trials or Max Dirtbike. A biker had to race to the finish line while maneuvering through obstacles, while trying to get the fastest time. We decided to use Pixi to display the game to the user as it was the one Professor St. Germain suggested. A users race data for each level would be stored into a database and displayed on their profile page or the leader board if they are on the top 3 players on that level. However we did not get this fully implemented so we just have a racer driving in a straight line and a pre seeded time built into their profile. We also planned on having a "ghost" of top racers, that could race alongside the user to try and beat them. To accomplish this, we planned on storing each racer's inputs into a database and loading it into the game when a user wanted to race against them. However we didn't get this implemented in time.

Along with gameplay we planned on adding in game cosmetics to satisfy more of the assignment standards. The shop was put in to allow the user to buy in game boosts to their speed and in game cosmetics to change the appearance of the bike. The shop is functional and a user that buys skins or nos will have it display on their profile, but we could not get these things to affect in game appearances or functionality. A user can also buy our in game currency. There is no way to actually buy the currency, but it just adds it to the user's profile for free. For these purchases we used AJAX to communicate with the controller so we could update the shop without refreshing the page. When a user presses the purchase button, it sends a post to the controller and depending on the outcome, it will send a sweet alert back to the user confirming or denying their purchase. We change different html values alongside sending the information to the controller to show our use of AJAX.

Our data is stored into a few simple databases. Our first one is a user table that stores all the relevant information that relates to a users, like shop purchases and their id. We display all of the user's information into a profile page. The second table is a leaderboards table that stores the user id, level id, and the users time. This information is displayed on the users profile page and on the leaderboards page if their time is low enough. We also scaffolded the built in user and roles identity database. We use identity to manage authorization and profile management for users. We used our user table in conjunction with the identity user table to store other information that would be important to users. The last database that we planned on having was saving a user's movements so we could display them as "ghosts" to race against them.

Feature Table:

Feature Name	Scope	Primary Programmer	Time Spent	File/Function	LoC
Inventory and buttons in store refresh with AJAX	UI	Jon & Giovanni	2 hrs / 2 hrs	shopPurchases.js	417
Shop interaction with DB	Back - end	Giovanni	2 hrs	UsersController	300
PIXI app gameplay	UI/Logic	Jon	8 hrs	Main.js	130
PIXI loading sprites and background scroll	UI	Jon	3 hrs	Background.js Bike.js Scroller.js	80
Game view	UI	Jon	2 hrs	index.cshtml	50
Shop view	UI	Giovanni	6 hrs	shop.cshtml	201
Highscore view	UI	Hank	1.5 hrs	LeaderBoard.cshtml	161
Profile view	UI	Giovanni	1.5 hrs	Profile.cshtml	160
DB initializer	Back - end	Hank	6 hrs	DBSeeder.cs	155
DB context	Back - end	Hank	2 hrs	DataBaseContext.cs	40

Models	Back - end	Hank	4 hrs	User.cs HighScore.cs	75
HighScore Controller	Back - end	Hank	1 hrs	HighScoresController.cs	150
Level Controller	Back - end	Hank	2 hrs	LevelsController.cs	160
Startup	Back-end	Hank	3 hrs	Startup.cs	120
Shop Controller	Back-	Jon	2 hrs	UsersController.cs	94

Individual Contribution:

Team Member	**Time Spent of Project**	**Lines of Code Committed**
Jon	25	628
Giovanni	25	948
Hank	30	1021

Jon:

My focus was adding the PIXI application to the web application. This proved to be much more difficult to implement than I had originally thought it would be. Part of the problem was that I began by following tutorials that did not lend themselves to the functionality needed for a side scroller. Once I found the information needed to create the side scrolling functionality, and to tile the background, I was able to make decent progress, but by then, time was limited, and we had many other more pressing issues that needed attention. I spent a good amount of time trying to help Hank fix database bugs, and helping Giovanni with the shop. I implemented a lot of AJAX functionality for the shop, such as modifying the database when skins were purchased, and updating the view so that skins could not be purchased more than once. The button appearance for the shop needed to be changed, and the button needed to be disabled as well.

Overall, I am disappointed with my ability to achieve the level of gameplay that I had anticipated and expected of myself. Time constraints and poor planning were my major issues. I underestimated the time that would be required to implement all of the features in the game that I wanted to, and I made poor choices in the beginning, which led to an initial implementation that would not work for our style of game. This required me to completely change the approach I took, and this unfortunately required too much time.

Hank:

My main focus was on creating the models for our database. I created a User, Bot, Level, and Highscore model. The User model contains important information related to the player such as their email, username, digital cash(gamerpoints), level unlocks, and which store items the user had purchased. The Bot model contains a link to access a file to the pre-recorded gameplay of the bot and an integer representing the time the bot took until crossing the finish line. The Level model contains the links for the files that would display the level and its obstacles. Lastly, the Highscore model keeps track of the file of each racer attempt at a level, time took until completion, LevelId, and user id

associated with it. These models allowed me to create a database and scaffold code fro the views and controllers. In addition to the models, I created the leaderboard page that would query the top 10 high scores from each level and display it for the user. Also, I enabled authentication for our web application and seeded the database with mock users, levels, high scores, bots, roles, users associated with those roles. After seeding, I had to test the website's functionality by retrieving from the local database and making sure the aws database was set up. Overall, I think I did an adequate job of fulfilling my tasks to create this web application. I was plagued with many errors from the start of the project that caused me to spend much more time than needed on material that was already covered in the class, like the creation of databases and authentication.

Giovanni:

My main focus for this project was on the views and controllers for our project. I implemented the HMTL for the Overview view, Profile view, and most of the Shop view. I also set up the controllers for each of these views. My main goal for this project was to create a Shop that used as many techniques, for making a web software project, that were taught to us by Professor St. Germain. For the Shop view I added AJAX to edit information in the database without refreshing the page along with changing and editing HTML values. For each button press in the Shop view, software will send a post to the controller and depending if a change was successful or not, it will return a sweet alert to the user based on the outcome. For all these views and for ones I did not work on I queried data from a database and displayed it to the user in a clean manner using different html structures and I added CSS to make the views look more appealing. I think I did a good job in accomplishing what was assigned to me, while also displaying knowledge and understanding of web software development.

Summary:

We believe that we performed “Adequate” for the time and effort that we put into the project. While we are missing some of the objectives that we planned on doing, those objectives ended up requiring too much time, so we could not implement them. Pixi ended up being difficult to create our game with, so we implemented a very basic game instead. We made this decision because we wanted to show more web software implementations that Professor St. Germain wanted to see from us. In addition, we had many struggles finding small bugs created by scaffolding, setting up the database, and minor authentication errors caused by preselecting authentication use when creating the project solution.

As a team we think we worked worked well together and did not have any personal conflicts with each other. One obstacle we had was that every team member was busy during the development cycle of our web application, so it was difficult to find time for us to work on it, or meet up. We believe that our project is worthwhile, because we demonstrated that we have learned web software development, MVC structure, server deployment, AJAX usage, database structure and creation, and proper team collaboration and construction. We hope that you see what we have accomplished the same way that we do.