PixelRunner Final Report

Christopher Lyons - u0970760
Emma Beatty - u1135862
Pavel (Pasha) Shestakov - u1097512

Abstract

PixelRunner was intended to be a single platformer game but has developed into a secure web application where players can play and review multiple games. The PixelRunner game itself is a single-player platformer game built using the Matter.JS physics engine. The game objective is to explore the Pixel world without dying by avoiding harmful objects, enemies, and projectiles. Users have a limited number of lives and can reach checkpoints that serve as re-spawning points. Users can also purchase skins through collecting coins in the game. Reviews can be added by users about any game in the web application and are displayed with each game.

VM URLs

Christopher Lyons: http://ec2-13-52-183-40.us-west-1.compute.amazonaws.com/
Emma Beatty: unable to host VM
Pasha Shestakov: http://ec2-18-208-168-234.compute-1.amazonaws.com/

Introduction

ASP Identity

Users must be authenticated before they are able to access and play any game, as user's IDs are tied to their save state in the database.

Character Controls

The player has WASD controls to move their character and SPACE to jump. The player can throw rocks at their current mouse location, climb ladders as well as interact with world objects with E. The game can be paused at any time by hitting ESC which will display the player's playtime statistics including both session and global playtime for each game.

Obstacles, Enemies, and Collision Detection

The game has obstacles and enemies to avoid. Lives are deducted when colliding with enemy projectiles or hazardous game world obstacles such as spikes or falling rocks.

Coin Collection and Store for Skins

The player can collect coins while playing the game that can be used to purchase skins within the game store. The player can then select which skin they'd like to use.

Game Sounds and Settings

The game sounds were mostly free, and included in the game which will be played on relevant event triggers. The sounds add an extra level of depth to the game. Also included in the game are settings for managing volume controls for both background music and sound effects. These are found by clicking the gear icon.

Game Reviews

The player can leave a written review and rating about the game, which is shown below the game. The player can edit their review and see other player's reviews.

Data Model

The data model is used to store game state as well as save states. The difference between the two is that game state is global for a specific <user, game> combination across all save states, it stores the amount of gold a player has available as well as the skins they have access to. The save state data model is for storing the number of lives a player has as well as their current checkpoint within the game.

Features

Feature Name	Scope	Primary Programmer	Time Spent	File/Function		LoC
Player controls (keybindings)	Game logic	Christopher Lyons	1 hour	Pixelrunner.js / characterControls()		77
Player motion (forces)	Game logic	Christopher Lyons	3 hours	Pixelrunner.js / move()		50
Game collision events	Game logic	Christopher Lyons	10 hours	pixelrunner.js / physicsEvents()		177
Update loop	Game logic	Christopher Lyons	15 hours	pixelrunner.js / physicsEvents()		204
Enemy creation	Game creation	Christopher Lyons	3 hours	pixelrunner.js / generate_enemies()		120
Overlay creation	Game UI	Christopher Lyons	1 hours	pixelrunner.js / initOverlay()		110
World initialization	Game creation	Christopher Lyons	5 hours	pixelrunner.js / createWorld()		410
Player projectiles	Game logic	Christopher Lyons	5 hours	pixelrunner.js / throw_projectile()		59
Pause/Unpause	Game logic	Christopher Lyons	1 hour	pixelrunner.js / pause/unpause		40
Animations	Game	Christopher	2 hours	pixelrunner.js /		50

	logic	Lyons		physicsEvents()		
Game initialization	Game logic	Christopher Lyons	2 hours	pixelrunner.js / init()		276
Basic store overlay	Game UI	Emma Beatty	4 hours	GameView.cshtml, game_controller.js, site.css		64
Store hover effects and asset creation	Game UI	Emma Beatty	5 hours	GameView.html, site.css, Game/Images		39
Refactored store hover effects	Game UI	Emma Beatty	2.5 hours	Game_controller.js, GameView.cshtml, site.css		51
Store grid initialization	Game UI	Emma Beatty	3 hours	Game_controller.js		30
Skin initialization, purchasing and selecting	Game UI	Emma Beatty	4 hours	Game_controller.js		51
Review Controller	Backend	Emma Beatty	3 hours	ReviewController.cs		110
Review UI and Ajax methods	Game UI	Emma Beatty	6 hours	GameView.cshtml, _Layout.cshtml, site.css		171
User and Review Seeding	Database	Emma Beatty	.5 hours	DbInitalizer.cs		67
Sweet Alerts / Game Font	Game UI	Emma Beatty	1 hour	GameView.cshtml, site.css, _Layout.cshtml, game_controller.js		40
Refactor physics_game.js to be an exported ES6 class	Front-end	Pasha Shestakov	7 hours	physics_game.js		265
New script that imports game module and loads the physics game (AJAX)	Front-end	Pasha Shestakov	2 hours	game_controller.js		37

Build UI for loading saved states	Front-end, UI	Pasha Shestakov	1 hour	Index.cshtml, site.css		34
Bug Fixes with data binding	Front-end	Pasha Shestakov	.5 hours	physics_game.js		8
Implement endpoints for loading game saves	Back-end	Pasha Shestakov	1 hour	GameController.cs		40
Make camera left/right movement smooth	Game UI	Pasha Shestakov	2.5 hours	physics_game.js		21
Add ability for multiple games	back-end	Pasha Shestakov	3 hours	GameController.cs, DbInitializer.cs, Game.cs		20
Add UI for multiple games, (Game selection page)	Front-end	Pasha Shestakov	5 hours	GameView.cshtml, Index.cshtml, _Layout.cshtml, site.css		307
Add models for GameContext,GameSkin, Review, UserGameSettings,UserGameState	Backend	Pasha Shestakov	3 hours	GameContext.cs, GameSkin.cs, Review.cs, UserGameSettings.cs, UsreGameState.cs		76
Add controller endpoint for getting UserGameSettings	Backend	Pasha Shestakov	1 hour	GameController.cs		22
Add controller endpoint for Updating UserGameSettings	Backend	Pasha Shestakov	1 hour	GameController.cs		24
Make sound settings saved in the database and	Backend/Frontend	Pasha Shestakov	6 hours	GameController.cs, UserGameSettings.cs,		279

loaded on page load, fix architecture to be able to load sounds without race condition				game_controller.js, gameSounds.js		
Implement UserGameState Backend endpoint, and front-end representation	Backend, Front-end		2 hours	GameController.cs		102

Individual Contribution

Team Member	Time Spent on Proj	Lines of Code Committed
Christopher Lyons	60 hours	2300
Emma Beatty	33 hours	800
Pasha Shestakov	35 hours	1,235

Contribution Overview: Christopher Lyons

Christopher Lyons created the game. This included integrating the usage of the matterJS physics library to manipulate rigid bodies through user input and animating sprites. In some cases, a lot of Christopher's time was spent actually modifying sprites and creating animations for resources that were once just a single image. Of course, another large area of Christopher's contribution within the game went into game logic, collision events, and world layout. Christopher also dedicated a portion of his time to finding suitable sound files to import and use for different game events. The overlay canvas displaying the HUD and pop-up messages were also part of Christopher's contributions over the past month. Furthermore, Christopher assisted other team members by redefining the existing database model for skins into something more practical, clear, and usable.

Contribution Overview: Emma Beatty

Emma's major contributions were creating the store for skins and the review feature. For the store, Emma spent time first constructing a basic UI element to display the available skins. She helped brainstorm a database model for the skins which was implemented by her teammates. Emma then used the skin database model to load the proper skins from the database, to load the user's selected skin, and then to facilitate selecting and buying specific skins. Throughout the process of creating the store--a fair amount of Emma's time was spent creating image assets for the store to match the game's overall feel. Emma's second major contribution was the review feature. The database model had been created for the reviews by Emma's teammates, but she took that model and set up the controller to handle the Review's CRUD functions. Emma also created the review UI and handled updating and displaying the reviews in real time in the view.

Contribution Overview: Pasha Shestakov

Pasha's major contributions were adding the ability for: multiple games, multiple user game saves for each game, saving user game settings such as volume, saving user game state such as lives, checkpoints, etc. To elaborate, Pasha wrote almost all of the controller end points and data models to accomplish the aforementioned, as well as hooking it up to the front-end. The front-end used es6 modules to organize the components, and had separate classes for user game settings and user game state that had the ability to update themselves via ajax and sync with the database. This allowed for volume settings, coins, lives, checkpoints,reviews, etc to be updated very easily from other parts of the game. Pasha was also the main designer behind all our data models, and also implemented them in code. This included managing the migrations. Pasha also built the page for game selection, which included a fancy css animation built in scss (sources online). Adding the ability for scss meant Pasha added a scss preprocessor to our project that also minified the css which allows for faster page load times.

Summary

PixelRunner has become an engaging and fun platformer game which helped our team realize more than just a game--but a functional web app. While the scope of the team's final project

was initially to create just a game, our team came together to create a web application where players can play and review multiple games including our very own unique platformer game.

Our team feels we've performed well and have produced a superior project that was able to effectively integrate many principles of Web Development. From utilizing ASP's Entity Framework to map a complex database, to using a physics engine, to using basic html and css--our site does it all. The main challenges our team overcame was deciding on an appropriate scope for our project and then realizing those decisions. We also struggled to find free time where we all were available but were able to overcome that through consistent communication via online messengers and group calls.

Our team not only conceptualized and produced an extensive game and web application that is usable now--but we also created a product with modularity and software engineering in mind. Our web app relies heavily on a database, and doesn't hardcode any of the data. We included the ability for multiple games to be added and have created the foundation for an application that could be further developed.