Team Blaze

# Team

Венцислав Конов – LucasBoy

Живка Стоева - Djufy

Илиян Йорданов - iliandji

Мирослав Гацанога - Miroslav\_Gatsan

Павел Христов - Lithical

Стефан Синапов - stefan.sinapov

# GitHub Repository

https://github.com/TeamBlaze/Hunting

# Summary

The genre of the game is а first person shooter, and it takes place in the Super Sonic Universe. The villain Eggman found a way to multiply himself, and Blaze, the main hero, was assigned a task to hunt down and kill as many Eggmen as possible. The game has a scoring system, and every Eggman awards Blaze with a point. The score system is global for all the players.

# Technologies

The two main programing languages used for the creation of the game are Java and JavaScript. Java is used for storing and retrieving high scores from a database server. JavaScript is used for creating drawings and animations using canvas and svg elements. The svg is used for creating the background of the game, and the canvas is used mainly for animations. HTML, CSS and Java Server faces are used for creating the web pages, and SQL queries are used for database access and manipulation.

# JavaScript

Object oriented principles are used for writing the JavaScript Code. The classes are divided into separate files. There is a main GameObject class who holds the common functions, and it is inherited by the two classes for the main heroes - Blaze and Eggman. There is a Contoller class responsible for handling mouse events. The renderer class was created, and it is responsible for handling animations and drawing on the screen. The Game class is the engine of the game, and it is responsible for the game logic. Almost every class has a CONGIG variable for holding the constants for its class.

# Java

Java classes are divided into Database Access, form beans and models and enumerations. Database Access Object supports two types of connection: connection pool and database driver.

# Java – JavaScript – Java Connection

When the page is first accessed the java bean connects to the database and selects all the scores, creates a string and sets this string to the value of an input field with hidden type. The page is rendered. When the rendering is done, JavaScript accesses the value of the hidden input, and it creates an array and renders the result on canvas.

When the game ends, JavaScript saves the high scores to the same hidden input field. A Java bean is bound to listen for changes of this value. When the change occurs, the bean gets the new value and compares it to the old value, and it saves the changes to the database.