GameBoard

All-in-one Arcade Games Website

Aleksandra Trifonova | Haemin Lee | Angelica Flores | Alexis Siguenza | Ian Iskra

Table of Contents

Introduction

Classic Arcade website which hosts a variety of classic games such as Pac-Man, Space Invaders, and Snake. These games are all simple in their gameplay and are able to be enjoyed by a variety of young and old people alike.

There are two portions to our project, which is the web page which was developed in react and bootstrap and Material UI/IO. The arcade games themselves were developed in Phaser, javascript and HTML.

For version control we used Github in order to keep track of the games and have easy access to prior version and be able to merge all of the parts that were worked on into one final project.

Framework : React.js

Why We Chose React?

- Newer emerging technology that's commonly used in front-end applications by industry.
- This is similar to React Native which we could use in app-development.
  - Gives us a stronger foundation to learn other secondary uses.
  - Can allow us to be more marketable in the job field.
- Easy to learn, information availability regarding documentation.
- It has bootstrap capability, @material-ui/icons (helpful libraries).

```
npx create-react-app website
npx install
npm install @material-ui/icons
npm install --save react-router-dom
npm install react-bootstrap bootstrap
```

Framework: Phaser.io

Why we chose Phaser.io?

- Fast rendering
- Supported by desktop and mobile
- Many built in features like collision and physics
- Beginner friendly tutorials
- Allows development of fully fledged games without having prior knowledge
- Phaser has three main functions that are vital to the life cycle; preload, create, and update.

Documentation

Official Docs

https://reactjs.org/docs/getting-started.html
React.js Official Documentation

https://reactjs.org/tutorial/tutorial.html
React.js Tutorial

https://photonstorm.github.io/phaser3-docs/
Phaser Official Documentation

http://phaser.io/learn
Phaser Tutorials and Examples

Community Resources

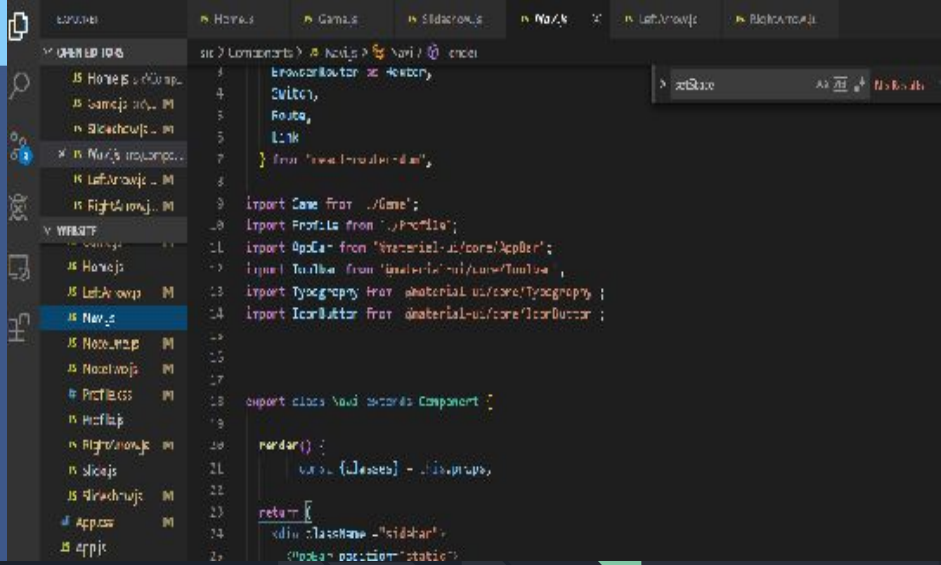
https://www.geeksforgeeks.org/bubble-sort/
Snake - used to check which space is occupied by the snake (how you die)

https://medium.com/@josephcardillo/using-math-random-in-javascript-c49eff920b11
Snake - used this function to get the food to be placed in random places on map

https://phaser.discourse.group/
Phaser community forums

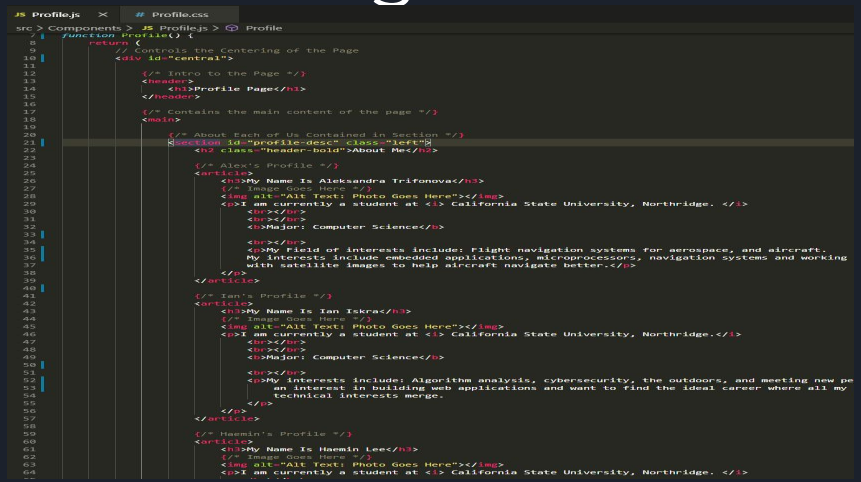
https://www.spriters-resource.com/
Game sprite assets

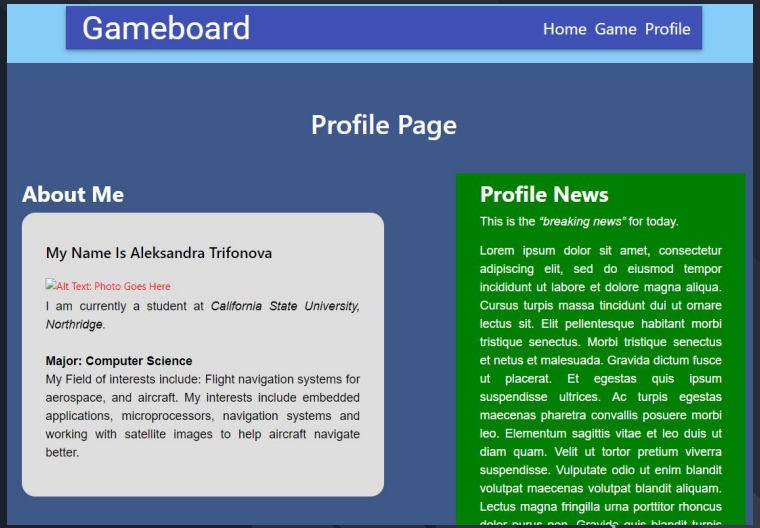
FEATURES



- Done by haemin, made Navigation, Home, Game Components, etc
- Used Material UI and Icons to make Navi Components
- Used bootstrap in card one and two, try to understand how to use bootstrap in react.js
- Used youtube and textbook to make slideshow and search information.
- Add some css in app.css to change some parts.

Profile Page

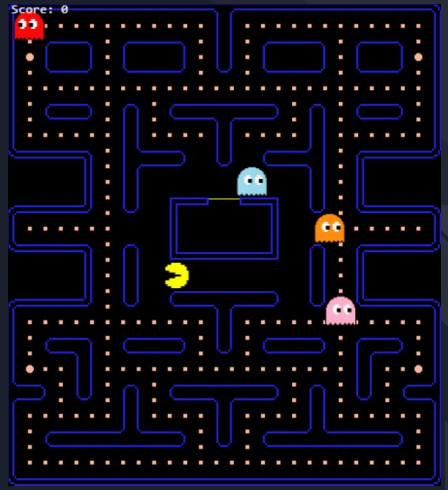




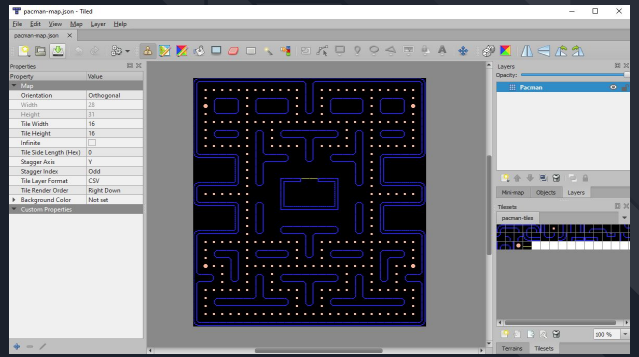
- Done by Ian Iskra, worked on the front end to display the HTML5 semantics within a React.js framework.
- The above navigation is carried across the website and Profile link will take you to this page.
- Semantics include: <main> contains the main content, <section> which contains <article> of each user profile, and <aside> which contains the News.
- Additionally, a class of .left and .right are applied to float in respective layouts.
- Within <article> *border-radius* style is applied for rounded edges and contains the information about user, divided by strategic headers (h2, h3,), paragraphs and appropriate line breaks (
).

Pac-Man Game

- Written in **Phaser 2** framework
  - Three major parts
    - Scene, Map, Sprites
  - Helpful Functionality
    - Asset handling
    - Keyboard control
    - Game physics
      - Velocity & collision
- **Pac-Man**
  - Control with arrow keys
  - Eat dots and kill ghosts
- **Ghosts**
  - Maze path-finding
  - “Afraid” run-away mode
- **Dots**
  - Super dots mode
  - Infinite respawn
  - High score count



```
preload: function () {

    // Dot assets
    this.load.image('dot', assetsDir + '/objects/dot.png');
    this.load.image('super-dot', assetsDir + '/objects/superdot.png');

    // Tile/map assets
    this.load.image('tiles', assetsDir + '/pacman-tiles.png');
    this.load.tilemap('map', assetsDir + '/pacman-map.json', null, Phaser

    // Sound assets
    this.load.audio('waka', assetsDir + '/sounds/waka.mp3');

    // Preload pacman assets
    this.pacman = new Pacman();
    this.pacman.preload(this);

    // Preload ghost assets
    this.inky = new Ghost('inky', {x: (13 * 16) + 8, y: (11 * 16) + 8});
    this.inky.preload(this);

    this.blinky = new Ghost('blinky', {x: 40, y: 40});
    this.blinky.preload(this);

    this.clyde = new Ghost('clyde', {x: 294, y: 224});
    this.clyde.preload(this);

    this.pinky = new Ghost('pinky', {x: 340, y: 280});
    this.pinky.preload(this);

},
```



Space Invaders Game

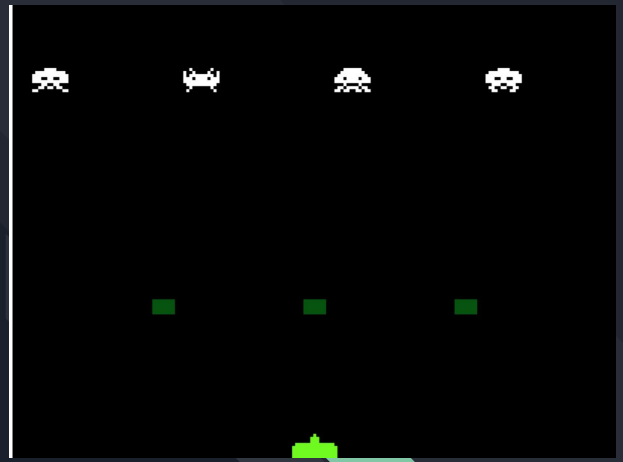
Written in Phaser,js version 3.

Main components in the code:

- Configuration () default code given by Phaser 3.
- Preload() assets load
- Create() positions for the sprites
- Update(): updates game every frame
- Many Global Variables *one very important shown

Other components:

- shootBullet() used velocity to move bullets -900 px up
- alienShoot() same as shootBullet only 900 px to move down
- randomAlien() makes aliens shoot randomly
- killIfOffscreen() so bullets won't bounce back



```
/* setting up our game asthetics and important functionalities*/
var config = {
    type: Phaser.AUTO,
    width: 800,
    height: 600,
    physics: {
        default: 'arcade',
        arcade: {
            gravity: { y: 0 }
        }
    },// end of physics
    scene: {
        preload: preload,
        create: create,
        update: update,


    }// end of scene

};//end of config
```

```
//creating variables for the game : player, leftkey and rightkey
var game = new Phaser.Game(config);
```

Space Invaders Preload()

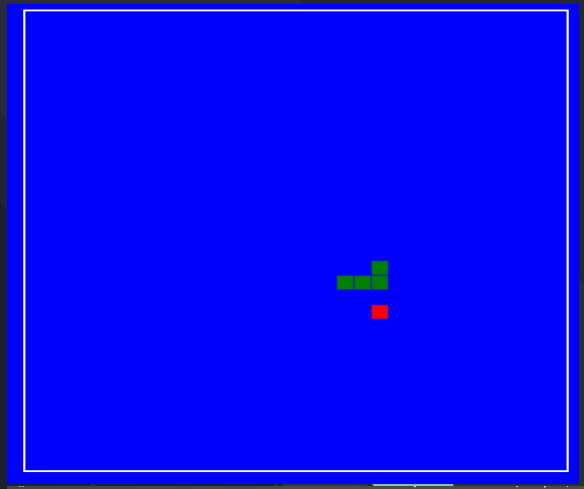
```
/*  where we load in external files (or “assets”)*/
function preload ()
{
    /* creating the sprite image of the ship in the preaload */
    this.load.image('player', 'assets/Ship.png');

    //blocks
    this.load.image('leftBlock', 'assets/FullBlock.png');
    this.load.image('middleBlock', 'assets/FullBlock.png');
    this.load.image('rightBlock', 'assets/FullBlock.png');

    //aliens
    this.load.image('alienOne', 'assets/InvaderA1.png');
    this.load.image('alienTwo', 'assets/InvaderB1.png');
    this.load.image('alienThree', 'assets/InvaderC1.png');
    this.load.image('alienFour', 'assets/InvaderA2.png');

    //bullets
    this.load.image('bullet', 'assets/Bullet.png');
    this.load.image('enemyBull', 'assets/Bullet.png');

}
```

Space Invaders Create()

```
 where we define the GameObjects that are necessary at the start of our game*/
nction create()

  let x = this.sys.canvas.width / 2;
  let y = this.sys.canvas.height;
  this.player = this.add.image(x,y, 'player').setOrigin(0.5, 1);
  this.player.depth = 9999;

  this.leftBlock = this.add.image(200, 400, 'leftBlock');
  this.middleBlock = this.add.image(400, 400, 'middleBlock');
  this.rightBlock = this.add.image(600, 400, 'rightBlock');

  this.alienOne = this.physics.add.image(100, 100, 'alienOne');
  this.alienTwo = this.physics.add.image(300, 100, 'alienTwo');
  this.alienThree = this.physics.add.image(500, 100, 'alienThree');
  this.alienFour = this.physics.add.image(700, 100, 'alienFour');
  alienArr = [this.alienOne, this.alienTwo, this.alienThree, this.alienFour];

  //setting the left key and right key ans pssce
  this.leftKey = this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.LEFT);
  this.rightKey = this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.RIGHT);
  this.spacebar = this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.SPACE);

  //bullets for spaceship
  bullets = this.physics.add.group();
  bullets.physicsBodyType = Phaser.Physics.ARCADE;
  bullets.enableBody = true;

  //group the bullets to load all at once
  for (var i = 0; i < 30; i++) {
      bullets.create(0.5, 1, 'bullet');
  }

  //makes the bullet invisble and puts it behinf the player ship
  bullets.children.each(function(bullet) {
      bullet.setActive(false);
      bullet.setVisible(false);
      //index for css and puts the bullet behind
      bullet.depth = 0; // always in back
  }, this);

  //bullets for ALIENS
  alienbull = this.physics.add.group();
  alienbull.physicsBodyType = Phaser.Physics.ARCADE;
  alienbull.enableBody = true;


  //group the bullets to load all at once ALIENS
  for (var i = 0; i < 30; i++) {
     alienbull.create(0.5, 1, 'enemyBull');

  }
```

Snake Game

- Done by Aleks, snake game developed in **html** and **javascript**
  - Main function eat the food, grow the snake
  - If the snake catches up to itself it dies (you lose)
- The reason the game was developed in html and javascript was because we wanted to make the comparison between making a game with phaser and in pure javascript and html to showcase that games can be developed in javascript **without** a framework and still render properly across **multiple platforms** tested in (safari, firefox, and **chrome**).
- Food
  - Made using math.random function and a bound checker to create the food and make sure the snake can eat the food.
- The part of the code displayed here are two of the main components
  - Keeping track of where the snake is and making sure it isn't backtracking on itself.
  - Using a bubble sort in order to keep track of which squares have collided into each other.
- Control with arrow keys
  - Automatic restart when you die.



indexsnakegame.html

```
  if (snake.y < 0) {
    snake.y = canvas.height - grid;
  }
  else if (snake.y >= canvas.height) {
    snake.y = 0;
  }
 //keeps track of where its been
  snake.cells.unshift({x: snake.x, y: snake.y});

  if (snake.cells.length > snake.maxCells) {
    snake.cells.pop();
  }
  context.fillStyle = 'red';
  context.fillRect(food.x, food.y, grid-1, grid-1);

  context.fillStyle = 'green';
  snake.cells.forEach(function(cell, index) {

    context.fillRect(cell.x, cell.y, grid-1, grid-1);

    if (cell.x === food.x && cell.y === food.y) {
      snake.maxCells++;
      food.x = getRandomInt(0, 25) * grid;
      food.y = getRandomInt(0, 25) * grid;
    }
    //simple bubble sort that keeps track of which squares collided into each other if at all.
    for (var i = index + 1; i < snake.cells.length; i++) {
      if (cell.x === snake.cells[i].x && cell.y === snake.cells[i].y) {
        snake.x = 160;
        snake.y = 160;
        snake.cells = [];
        snake.maxCells = 4;
        snake.dx = grid;
        snake.dy = 0;
        food.x = getRandomInt(0, 25) * grid;
        food.y = getRandomInt(0, 25) * grid;
      }
    }
  });
}
//event listener for the keystrokes that are used for gameplay
document.addEventListener('keydown', function(e) {
```

Challenges

- **Integrating Phaser with React**
  - Both have a steep learning curve
  - DOM (React) vs. Canvas (Phaser)
- **Proper Github etiquette and version control**
  - Keeping track of old versions, which worked and which didn't and proper pull and push commands (everyone had a different machine and was using git in a different way)
- **Variable scope in Javascript**
  - Dynamic objects and states
  - components [React], scenes [Phaser]
- **Framework Abstraction**
  - Phaser does a lot of behind the scenes work that makes it difficult to understand where certain things are coming from
  - Can make debugging difficult - Single changes can break code.
- **NPM dependencies**
  - Install errors, merge conflicts
- **Material UI**
  - Attempting to make Material UI and icons work with React
- **Short Time Frame**
  - Learning the frameworks of phaser and react quickly


DEMO

Lessons

- Version control using github
- Creating UI components in react
- Make as many components as modular as possible
  - Page Components
    - Profile, Home, Navigation, Notes
  - Basic modular Phaser model
    - Pac Man - Ghosts, Pac-man, dots
    - Space Invaders - Bullets, Aliens, and the Ship
    - Snake - Food function (randomly shows up)
- Getting the frameworks and developer environment to work in different operating systems

Live Website

Want to play on your own browser? Go to the link below!

https://siguenza.net:3001/


That's all Folks!