Phaser JS Game – Keuzedeel Mobile Application Development.
“Bunny Jumper”
https://julianvh.nl/keuzedeel/

Door *Julian van Husen*

Inleiding

Aan het begin van dit keuzedeel ben ik begonnen met een game. Ik had geen idee waar ik moest starten. Toen raadde Martijn Kunstman (mijn docent), mij een website aan waar je een pdf kon krijgen waarin stond hoe je een game kon maken met Phaser 3. Ik heb de hele tutorial gevolgd en zelf ook elementen toegevoegd! Het was een erg lastig proces omdat er ook niet zoveel *goede* documentatie te vinden was over het game framework. In dit verslag zal ik mijn ontwikkelomgeving en eindproduct beschrijven in nauw detail.

Inhoudsopgave

De ontwikkelomgeving.

de ontwikkelomgevingen die ik gebruikt heb is Visual Studio Code. Visual Studio Code heb ik gebruikt om mijn applicatie mee te coderen.

Kenmerken van de ontwikkelomgeving.

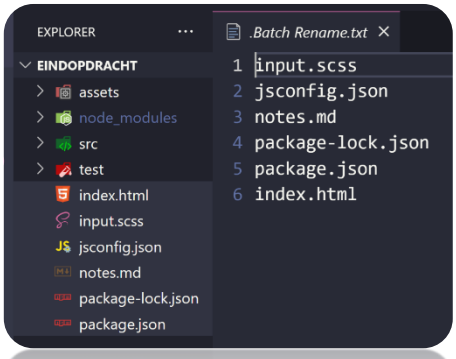
VSCode is een open-source code-editor. Maak door al zijn extensies en mogelijkheden kun je het gerust een IDE noemen.

Zoals ik al eerder heb vermeld heb ik voor het maken van deze applicatie Visual Studio Code gebruikt. Een van de voornaamste redenen dat ik VSCode gebruik is de mogelijkheid om je eigen snippets te kunnen maken! Hierdoor kun je geleidelijk je eigen “shortcut-database” opzetten. VSCode heeft hier een geweldige syntax voor! Hiernaast een aantal van mijn zelfgemaakte snippets.

Wat ook heel erg fijn is is dat je een apart JSON bestand hebt waarin je de instellingen van VSCode tot in de details kunt aanpassen. Zo kun je bijvoorbeeld aanzetten dat je in verschillende bestandsextensies snippets in strings aangeboden krijgt…

Heel handig als je bijvoorbeeld een framework als bootstrap of tailwind gebruikt. Dan hoef je niet steeds weer op “ctrl+space” te drukken om je snippets tevoorschijn te toveren.

Ook heeft VSCode een handig systeem met extensies. Je kunt “on-the-fly” extensies downloaden, aanzetten, uitzetten, combineren en instellingen aanpassen. Ook kun je ervoor kiezen om sommige extensies alleen te gebruiken in verschillende bestandsextensies. Extensies kunnen bijvoorbeeld je “keybindings” veranderen, extensies kunnen je ook snippets geven als je in een specifiek bestand/codeblock zit. Extensies kunnen zelfs liveservers zijn of een tool waarmee je in de vorm van een tekstbestand een grote groep bestanden een andere naam kunt geven.

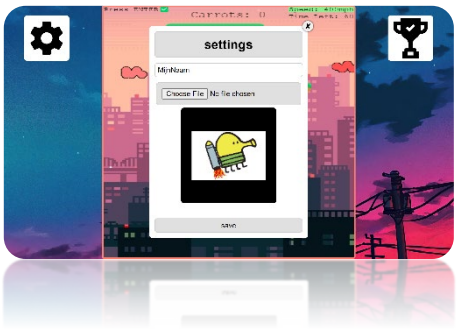


En dan als laatste hebben we nog de mogelijkheid om heel veel handige built-in shortcuts te gebruiken. Van multi-cursor editing naar het dupliceren van regels code. Naar zelfs een hele syntax genaamd emmet toegeweid aan het sneller aanmaken van html-elementen!

VScode is “by-far” de beste code editor die er bestaat voor web-development.
Dit is de rede dat ik VSCode gebruik. Ik kan de instellingen volledig veranderen totdat het aan mijn behoeften voldoet.

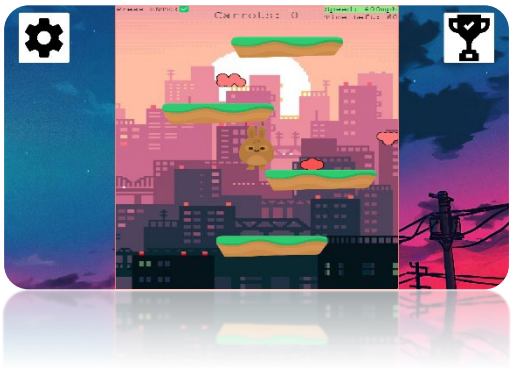
Het ontwikkelproces.

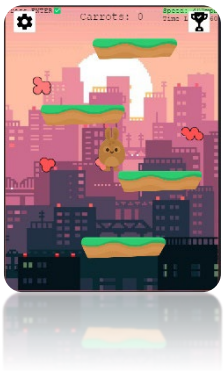
Zoals eerder aangegeven heb ik een game gemaakt met het JavaScript Framework: “Phaser JS”. Ik heb eerst een geschreven tutorial gevolgd (deze tutorial is te vinden in de zip). Daarna heb ik verder elementen toegevoegd zoals: Een scoreboard met een modal, game-timer, start game screen en een instellingenknop met een modal waarin je je naam kunt veranderen en waarin je een foto kunt uploaden, deze foto wordt dan het karakter. Als je géén foto uploadt dan krijg je de default-bunny toegewezen als karakter. Daarnaast heb ik ook een parallax ingebouwd, op de wolken en op de achtergrond.

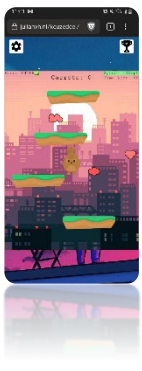


De aansluiting

Mijn game sluit perfect aan op m’n concept. De game wordt gespeeld op een normale aspect ratio wanneer het op een mobielscherm wordt gespeeld en wanneer het op een desktop wordt gespeeld wordt er een 1:1 aspect ratio gebruikt. De Game beschikt ook over de mogelijkheid om de accelerometrie meter van je mobiele telefoon te gebruiken. je kunt het karakter geheel besturen met het draaien van je telefoon!

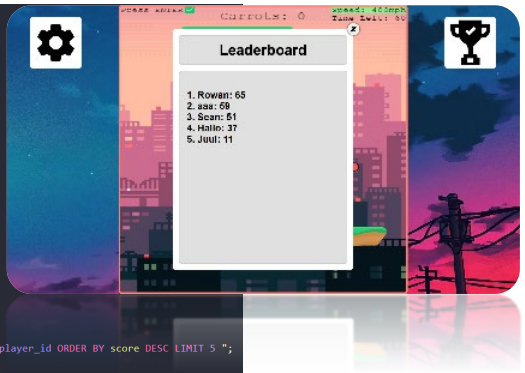






Wel heb ik het concept iets wat uitgebreid. Nu kan je veel hoger springen en wordt je snelheid steeds aangepast. Dit geeft een wat meer uitdagendere speelervaring. Verder heb ik net zoals beschreven ook een database toegevoegd!

```php
<?php
include "./db_conn.php";
$obj = new Connection();
$conn = $obj->connection();

// Checks if there's a GET request being made
if (isset($_GET['player']) && isset($_GET['score'])) {
    $player = $_GET['player'];
    $score = $_GET['score'];

    $my_date = date("Y-m-d H:i:s");

    // inserts into the player table... Name and Date
    $query = " INSERT INTO `player` (name , date) VALUES('$player', '$my_date') ";
    $result = mysqli_query($conn, $query);

    $player_id = $conn->insert_id;
    // also inserts into the score table
    $query = " INSERT INTO `score` (player_id, score, playdate) VALUES('$player_id', '$score', '$my_date') ";
    $result = mysqli_query($conn, $query);
}

// uses a join to combine the two tables together and limits the results to 5
$query = " SELECT player.name AS player, score.score AS score FROM player INNER JOIN score ON player.id = score.player_id ORDER BY score DESC LIMIT 5 ";
$result = mysqli_query($conn, $query);

?>
<!-- Just renders all the results to the webpage, so the JavaScript fetch can pull the data  -->
<?php
while ($row_users = mysqli_fetch_array($result)) {
    echo "<li>" . ($row_users['player']) . ": " . ($row_users['score']) . "</li>";
}

// closes the connection
$conn->close();
exit();
?>
```



Wat wel een dingetje is om te benoemen is het feit dat alles achter de schermen gebeurd. Het schrijven en ophalen van gegevens gebeurt geheel in JavaScript met een fetch.

```
345   saveDataToDatabase() {
346     // sends the data towards the PHP file
347     fetch(
348       `${this.path}?player=${username.value}&score=${this.carrotsCollected}`,
349       {
350         method: "get",
351       }
352     )
353       .then(function (response) {
354         if (response.status >= 200 && response.status < 300) {
355           return response.text();
356         }
357         // throws an error if the request could not be made
358         throw new Error(response.statusText);
359       })
360       .then(function (response) {
361         // renders the result to the leaderboard
362         leaderboardData.innerHTML = response;
363       });
364   }
365
366   retrieveDataFromDatabase() {
367     // gets the data from the PHP file. Doesn't insert anything
368     fetch(`${this.path}`, {
369       method: "get",
370     })
371       .then(function (response) {
372         if (response.status >= 200 && response.status < 300) {
373           return response.text();
374         }
375         throw new Error(response.statusText);
376       })
377       .then(function (response) {
378         leaderboardData.innerHTML = response;
379       });
380   }
381 }
```

Verder sluit de game ook goed aan op m'n Functioneel en Technisch ontwerp. Ik heb de "gear" en "throphy" icon verwerkt in mijn game en zijn er "tiles" waarop het karakter kan springen. De game heeft een duidelijk begin en einde. Nogmaals, je kunt een foto uploaden, en de game maakt gebruik van de accelerometrie meter in je telefoon. Tevens beweegt de achtergrond ook mee.

Helaas heb ik niet de kleurenpaletten gebruikt die in het verslag stonden. Wel heeft de game soortgelijke aansluitend kleuren. Ook het lettertype is meer "retro". wat opzich wel een mooi effect geeft aan de game.

Code voorbeelden

In het schrijven van de applicatie heb ik uitsluitend gewerkt met JavaScript. Hieronder een aantal code voorbeelden.

Hieronder staat een functie die luistert naar de bewegingen van de telefoon. Hij pakt het event op en kijkt naar hoeveel graden de telefoon gekanteld wordt. Is dat meer of minder dan 60 graden dan wordt het karakter naar links of naar rechts bewogen. Anders springt het karakter in een rechte lijn naar boven. Ook worden in de functie de wolken en de achtergrond meebewogen naar aanleiding van de kant waarop het karakter beweegt.

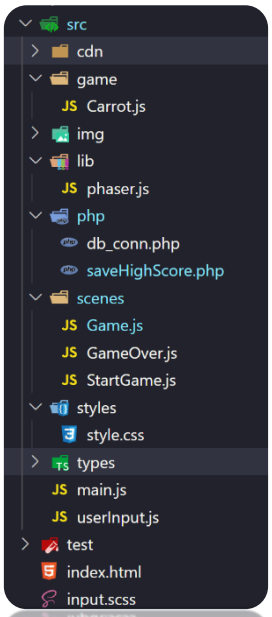
```
window.addEventListener("devicemotion", (e) => {
  const x = Math.round(e.accelerationIncludingGravity.x) * 30;
  if (x < -60 && !touchingDown) { // right
    this.player.setVelocityX(200);
    this.background.tilePositionX -= 0.0004;
    this.clouds.tilePositionX -= 0.0005;
  } else if (x > 60 && !touchingDown) { //left
    this.player.setVelocityX(-200);
    this.background.tilePositionX += 0.0004;
    this.clouds.tilePositionX += 0.0005;
  } else {
    this.player.setVelocityX(0);
    this.clouds.tilePositionX += 0.002;
  }
}, true);
```

Hiernaast een stuk code die ervoor zorgt dat de engine die wordt gebruikt (in dit geval de arcade physics engine) luistert naar “collisions”. Ook staan er in de code functies gedefinieerd als: sprite, setScale, startFollow, setDeadzone enzovoorts. Al deze functies zijn functies die komen van de Phaser CDN. Deze functies zorgen ervoor dat de gebruiker makkelijker elementen kan implementeren en zich geen zorgen hoeft te maken over collisions en camera’s.

```
// Bunny🐇 Physics
this.player = this.physics.add
  // adds the bunny sprite with the width and height
  .sprite(240, 320, "bunny-stand")
  .setScale(0.5);
// adds the collider on the platfroms and the player
this.physics.add.collider(this.platforms, this.player);

// sets the collision detection only on the bottom of the bunny
this.player.body.checkCollision.up = false;
this.player.body.checkCollision.left = false;
this.player.body.checkCollision.right = false;

// camera follows the player
this.cameras.main.startFollow(this.player);
// Locks the background and makes sure the player can go out
of the screen by a little margin
this.cameras.main.setDeadzone(this.scale.width * 1.5);
this.carrots = this.physics.add.group({
  classType: Carrot,
});
```

Persoonlijke toevoeging

het was een hele klus om deze opdracht te maken. Vooral door het feit dat Phaser een slecht gedocumenteerd framework is met weinig resources. Een hoop kwam neer op trial and error. Maar voor de rest heb ik er echt heel erg veel van geleerd. Vooral ook omdat de game, object georiënteerd geprogrammeerd is met een aantal functies. Maar ook al kijkend naar de “folder-structure” zegt al een hoop over de complexiteit van de game.



Wat wel zo is aan Phaser, is dat als je de moeite erin steekt om het framework echt te leren. Het echt een heel mooi framework is om games te maken. Hieronder een voorbeeld waarom het zo’n handig is. Het framework maakt gebruik van scenes. Deze scenes staan gedefinieerd in de “scene” array. Het mooie aan Phaser is dat je verschillende schermen ofwel “scenes“ Hebt die verschillende dingen uitvoeren waardoor je eigenlijk meerdere “schermen” hebt die je heel snel over elkaar heen kan leggen. En allemaal doen ze wat anders. Hartstikke handig als je bijvoorbeeld levels wilt maken in je game. Of als je zoals ik gedaan hebt een start-game en game-over screen wilt hebben. En dit alles in een canvas. Het is ongelofelijk.

```
// imports the necessary files and classes
import Phaser from "./lib/phaser.js";
import Game from "./scenes/Game.js";
import StartGame from "./scenes/StartGame.js";
import GameOver from "./scenes/GameOver.js";

// makes sure the game will be configured
export default new Phaser.Game({
  // sets WebGL or Canvas based on browser compatibility
  type: Phaser.AUTO,
  // width / height of the game screen
  width: 480,
  height: 640,
  // loads the different child classes (in order)
  scene: [StartGame, Game, GameOver],
  physics: {
    // declares which physics engine will be used
    default: "arcade",
    arcade: {
      gravity: {
        // sets the amount of gravity
        y: 200,
      },
      // shows lines and hitboxes handy for development
      debug: true,
    },
  },
  fps: {
    // sets the framerate
    target: 60,
    forceSetTimeOut: true,
  },
});
```