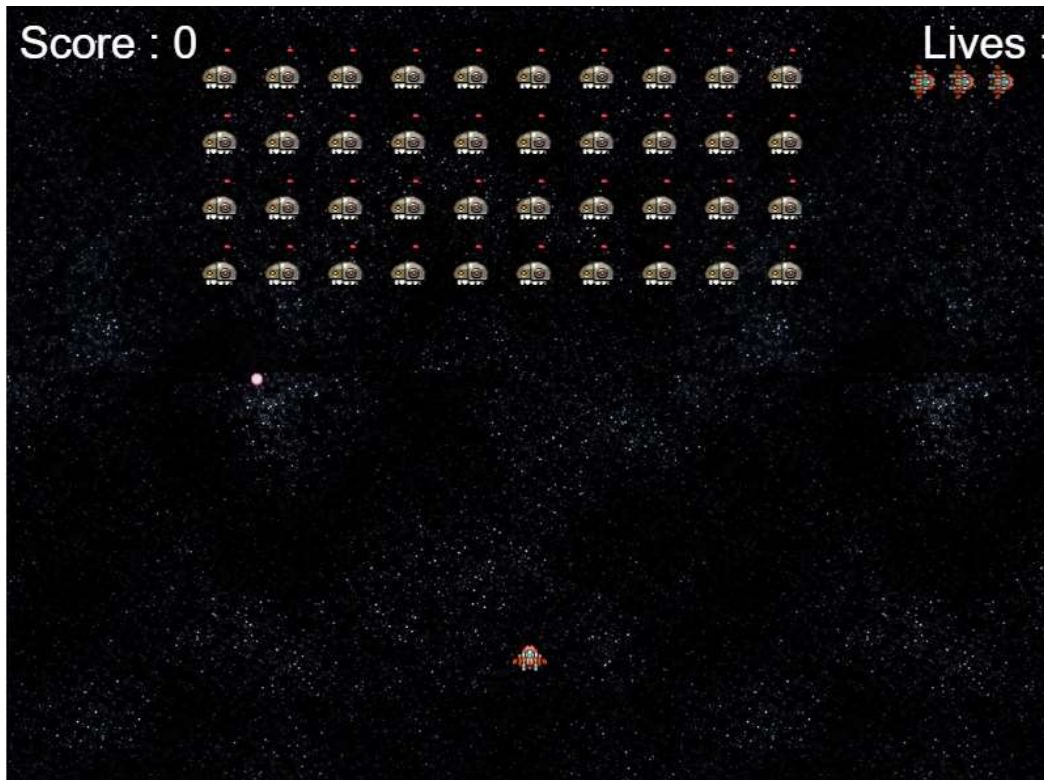


SPACE INVADERS



Het spel Space Invaders is een klassieker. Het spel is 40 jaar oud! Maar nog steeds leuk en je kunt het ook zelf maken. In deze workshop gaan we aan de slag met Phaser.

Wat is Phaser?

De eerste vraag is misschien: wat is dat eigenlijk, Phaser?

Phaser is een verzameling functies, gemaakt in JavaScript, waarmee het heel makkelijk is om games te maken die in de browser werken. Omdat al die functies door andere mensen al gemaakt is, hoef je die zelf niet meer te bedenken. Als je meer wilt weten over Phaser ga je naar de website phaser.io.

Wat heb je nodig?

Om te beginnen heb je de volgende dingen nodig:

1. Een browser (duh). Dit is bijvoorbeeld Chrome, Firefox of Edge.
2. Een programma om code te schrijven. Wij gebruiken hiervoor Visual Studio Code.
3. Een webserver om de bestanden van je spel naar de browser te sturen. Wij gebruiken het programma Fenix zodat je alles op je eigen computer kunt doen.

Browser

Een browser staat standaard al geïnstalleerd. Het maakt niet uit welke je gebruikt. Moderne browsers zijn prima geschikt voor spelletjes.

Visual Studio Code

Voor het schrijven van JavaScript code kan je notepad (of kladblok) gebruiken, maar erg prettig is dat niet. We gebruiken liever Visual Studio Code. Ook dit programma staat al geïnstalleerd, maar je kunt het ook gratis downloaden op: <https://code.visualstudio.com/>


Webserver

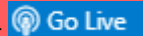
Een webserver zorgt ervoor dat de browser het spel kan openen. Vroeger kon je die bestanden ook vaak direct vanaf de computer openen, maar de makers vonden dat niet zo veilig. Je kon er allerlei rare dingen mee doen, en virussen mee oplopen. Dat willen we natuurlijk niet!

Je kunt in Visual Studio Code ook een webserver gebruiken. Je installeert hem zo:

Druk in Visual Studio Code op CTRL+P en type:

```
ext install ritwickdey.liveserver
```

Druk op ENTER om de installatie te starten. Als het klaar is, zie je rechtsonder in je scherm: .

LET OP: Je ziet  *alleen* wanneer je in Visual Studio Code een folder hebt geopend. Je opent een folder met CTRL-K + CTRL-O (dus CTRL indrukken en dan op O en K drukken). Het kan ook via het menu: File > Open Folder. De folder die je kiest is de bron voor de webserver.

Je spel kan je starten door op Go Live te klikken... als je klaar bent met programmeren.

De onderdelen van het spel

Voor het gemak hebben we map gemaakt waar al wat in staat. Start Visual Studio Code en kies **File > Open Folder**. Ga naar de map `\javascript\opdrachten\invaders` kies **Select Folder**.

Hier zie je bestanden **index.html** en **invaders.js**. In **index.html** staat dit:

```
<html>
<head>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/phaser/2.6.2/phaser.min.js"></script>
  <script src="invaders.js"></script>
</head>
<body>
</body>
</html>
```

Het bestand **index.html** is alleen een plek om spel te kunnen open in een browser. Het echte werk zit in het bestand **invaders.js**.

Maar dat bestand is nu nog leeg.

De drie belangrijkste functies die een spel nodig heeft zijn **preload**, **create** en **update**. We gaan elke functie invullen, maar we beginnen met de variabelen die we nodig hebben in het spel.

Variabelen

Variabelen zijn plekken in het geheugen van de computer. Je mag zelf een naam verzinnen voor zo'n plek, maar het is wel handig dat de naam duidelijk maakt wat je er in wilt bijhouden.

We hebben veel variabelen nodig om de status van het spel bij te houden. Zo moeten we de score bijhouden, de plek waar jouw ruimteschip staat, hoeveel aliens er zijn en hoeveel levens je over hebt. In JavaScript kan je variabelen definiëren met de term **var**. Er zijn verschillende soorten variabelen: getallen, teksten, lijsten en waar/onwaar. JavaScript doet trouwens niet zo moeilijk over die soorten. Je kan het door elkaar gebruiken. Dat is handig, maar kan voor jou als programmeur soms verwarrend zijn. Let dus zelf goed op welke soort waarde, een getal of een tekst bijvoorbeeld, je in een variabele stopt.

De hele lijst variabelen staat hieronder. Voeg deze toe aan **invaders.js**.

```
var player;
var aliens;
var bullets;
var bulletTime = 0;
var cursors;
var fireButton;
var explosions;
var starfield;
var score = 0;
var scoreString = '';
var scoreText;
var lives;
var enemyBullet;
var firingTimer = 0;
var stateText;
var livingEnemies = [];
```

game

Er ontbreekt nog één variabele: **game**. Deze voeg je nu toe.

```
var game = new Phaser.Game(800, 600, Phaser.AUTO, 'phaser-invaders',
{ preload: preload, create: create, update: update });
```

Je ziet dat de variabele **game** gevuld wordt met gegevens over het spel. Het soort variabele is bijzonder: **Phaser.Game**. Dat is een speciale soort dat je alleen kunt gebruiken omdat we **Phaser** gebruiken. In het html-bestand (**index.html**) zie je een verwijzing naar Phaser en daarom weet de browser wat er allemaal kan en mag met deze variabele.

preload

De functie **preload** zorgt ervoor dat alle plaatjes in het geheugen van de computer worden geladen. Als een spel ook geluid maakt wordt ook geluid geladen. Spellen als Minecraft, Fortnite, GTA en CoD hebben zo veel plaatjes en geluid dat het laden soms best lang duurt. Onze space invaders is niet zo groot, dus het laden is heel snel.

Voeg de volgende code toe aan **invaders.js**.

```
function preload() {  
    game.load.image('bullet', 'assets/bullet.png');  
    game.load.image('enemyBullet', 'assets/enemy-bullet.png');  
    game.load.spritesheet('invader', 'assets/invader32x32x4.png', 32, 32);  
    game.load.image('ship', 'assets/player.png');  
    game.load.spritesheet('kaboom', 'assets/explode.png', 128, 128);  
    game.load.image('starfield', 'assets/starfield.png');  
    game.load.image('background', 'assets/background2.png');  
}
```

De plaatjes staan in de map **assets** onder de map waar je spelcode staat. Elk plaatje wordt met deze code in het geheugen geladen. Verderop in het spel kan je nu makkelijk verwijzen naar deze plaatjes door de naam die we aan elk plaatje hebben gegeven, zoals **bullet**, **ship** en **kaboom**.

create

Na de functie **preload** komt de functie **create**. Als **create** wordt uitgevoerd krijgen alle onderdelen van het spel een plek op het scherm. Het spel krijgt een achtergrond, er zijn aliens en explosies. En je eigen ruimteschip komt erop te staan.

De functie **create** begin je zo.

```
function create() {  
  
  
}
```

De functie bevat nu nog geen instructies. Die ga je nu toevoegen. We beginnen met het spelsysteem dat we gebruiken. Voeg deze code toe *in de functie*.

```
game.physics.startSystem(Phaser.Physics.ARCADE);
```

Met Phaser.Physics.ARCADE geven we door dat je in het spel gebruik kan maken van zwaartekracht zodat dingen naar beneden kunnen “vallen” en dat ze tegen elkaar kunnen botsen.

Dus nu ziet de functie **create** er zo uit.

```
function create() {  
  
    game.physics.startSystem(Phaser.Physics.ARCADE);  
  
}
```

Alle volgende regels komen ook *in de functie* te staan.

```
// De achtergrond van het spel  
starfield = game.add.tileSprite(0, 0, 800, 600, 'starfield');  
  
// De laser kogels van onze held  
bullets = game.add.group();  
bullets.enableBody = true;  
bullets.physicsBodyType = Phaser.Physics.ARCADE;  
bullets.createMultiple(30, 'bullet');  
bullets.setAll('anchor.x', 0.5);  
bullets.setAll('anchor.y', 1);  
bullets.setAll('outOfBoundsKill', true);  
bullets.setAll('checkWorldBounds', true);  
  
// De laser kogels van de vijand  
enemyBullets = game.add.group();  
enemyBullets.enableBody = true;  
enemyBullets.physicsBodyType = Phaser.Physics.ARCADE;  
enemyBullets.createMultiple(30, 'enemyBullet');  
enemyBullets.setAll('anchor.x', 0.5);  
enemyBullets.setAll('anchor.y', 1);  
enemyBullets.setAll('outOfBoundsKill', true);  
enemyBullets.setAll('checkWorldBounds', true);  
  
// Onze held!  
player = game.add.sprite(400, 500, 'ship');  
player.anchor.setTo(0.5, 0.5);  
game.physics.enable(player, Phaser.Physics.ARCADE);  
  
// De slechteriken  
aliens = game.add.group();  
aliens.enableBody = true;  
aliens.physicsBodyType = Phaser.Physics.ARCADE;  
  
createAliens();
```

```

// De score
scoreString = 'Score : ';
scoreText = game.add.text(10, 10, scoreString + score,
    { font: '34px Arial', fill: '#fff' });

// Levens
lives = game.add.group();
game.add.text(game.world.width - 100, 10,
    'Lives : ', { font: '34px Arial', fill: '#fff' });

// Plek voor teksten
stateText = game.add.text(game.world.centerX, game.world.centerY, ' ',
    { font: '84px Arial', fill: '#fff' });
stateText.anchor.setTo(0.5, 0.5);
stateText.visible = false;

for (var i = 0; i < 3; i++)
{
    var ship = lives.create(game.world.width - 100 + (30 * i), 60, 'ship');
    ship.anchor.setTo(0.5, 0.5);
    ship.angle = 90;
    ship.alpha = 0.8;
}

// Het explosie effect
explosions = game.add.group();
explosions.createMultiple(30, 'kaboom');
explosions.forEach(setupInvader, this);

// Toetsen om het spel mee te spelen
cursors = game.input.keyboard.createCursorKeys();
fireButton = game.input.keyboard.addKey(Phaser.Keyboard.SPACEBAR);

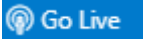
```

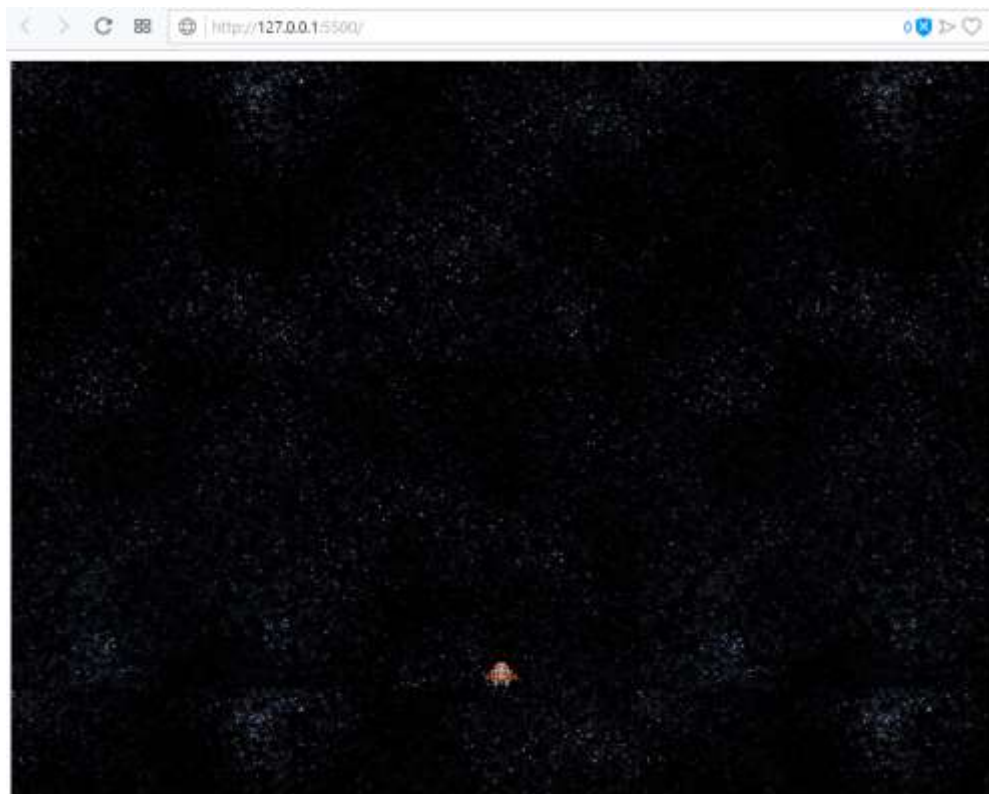
Zo dat was een flink lap code. Kijk je ook naar wat je hebt ingevoerd?

Ok, we zijn bijna zover dat we wat op het scherm kunnen krijgen. Voeg alleen de functies **update** en **setupInvader** toe. Die vullen we later met instructies.

```
function update()
{
}
function setupInvader()
{
}
```

Zorg dat je alles goed bewaart met CTRL-S (of **File > Save**).

Klik nu rechtsonder in je scherm op . Als je geen fouten hebt gemaakt, zie je het veld van het spel.



De ruimte ziet er nog een beetje leeg uit. Onze held staat klaar om te vuren, maar de aliens zijn er nog niet.

Ook daar hebben we een functie voor.

```

function createAliens () {

    for (var y = 0; y < 4; y++)
    {
        for (var x = 0; x < 10; x++)
        {
            var alien = aliens.create(x * 48, y * 50, 'invader');
            alien.anchor.setTo(0.5, 0.5);
            alien.animations.add('fly', [ 0, 1, 2, 3 ], 20, true);
            alien.play('fly');
            alien.body.moves = false;
        }
    }

    aliens.x = 100;
    aliens.y = 50;

    // De aliens bewegen van links naar rechts
    var tween = game.add.tween(aliens).to( { x: 200 }, 2000,
        Phaser.Easing.Linear.None, true, 0, 1000, true);

    // De aliens komen steeds dichterbij
    tween.onRepeat.add(descend, this);
}

```

De functie **createAliens** maakt 4 rijen van 10 aliens. Ook komen de aliens steeds dichterbij. Voor het dichterbij komen hebben we een aparte functie.

```

function descend() {

    aliens.y += 20;

}

```

Je kunt natuurlijk zelf kiezen hoe snel of hoe langzaam de aliens dichterbij komen. Het spel moet niet te makkelijk zijn, maar ook niet te moeilijk.

update

De functie **update** hadden we al toegevoegd om wat te kunnen zien. Maar er stond nog niks in. De functie **update** wordt 60 keer per seconde uitgevoerd. We gebruiken deze functie om de status van het spel bij te werken.

```
function update() {

    // Laat de achtergrond rollen. Zo krijg je een vlieg-effect
    starfield.tilePosition.y += 2;

    if (player.alive)
    {
        // De speler beweegt alleen als je de cursor toetsen indrukt
        player.body.velocity.setTo(0, 0);

        if (cursors.left.isDown)
        {
            player.body.velocity.x = -200;
        }
        else if (cursors.right.isDown)
        {
            player.body.velocity.x = 200;
        }

        // Vuur je een laser kogel af?
        if (fireButton.isDown)
        {
            fireBullet();
        }

        // De vijand kan ook vuren
        if (game.time.now > firingTimer)
        {
            enemyFires();
        }

        // Als je ergens tegenaan botst ben je af.
        game.physics.arcade.overlap(bullets, aliens, collisionHandler, null, this);
        game.physics.arcade.overlap(enemyBullets, player, enemyHitsPlayer, null, this);
    }
}
```

In de functie **update** gebruiken we functies die we nog niet gemaakt hebben. We moeten nu de volgende functies maken:

- fireBullet

- enemyFires
- collisionHandler
- enemyHitsPlayer

We beginnen met **fireBullet**.

```
function fireBullet () {  
  
    // Er zit een grens op hoe snel je achter elkaar mag vuren.  
    if (game.time.now > bulletTime)  
    {  
        // Haal een kogel op  
        bullet = bullets.getFirstExists(false);  
  
        if (bullet)  
        {  
            // Vuur!  
            bullet.reset(player.x, player.y + 8);  
            bullet.body.velocity.y = -400;  
            bulletTime = game.time.now + 200;  
        }  
    }  
}
```

Zie je ook welke getal je moet veranderen om sneller of langzamer te vuren?

De slechteriken mogen ook vuren. Dat gebeurt in de functie **enemyFires**.

```

function enemyFires () {

    // Haal een kogel op
    enemyBullet = enemyBullets.getFirstExists(false);

    livingEnemies.length=0;

    aliens.forEachAlive(function(alien){

        // Zet alle aliens die nog leven in een lijst
        livingEnemies.push(alien);
    });

    if (enemyBullet && livingEnemies.length > 0)
    {
        var random=game.rnd.integerInRange(0,livingEnemies.length-1);

        // selecteer één willekeurige alien die mag schieten
        var shooter=livingEnemies[random];
        // schiet de kogel af van deze alien
        enemyBullet.reset(shooter.body.x, shooter.body.y);
        // zorg dat de kogel richten onze held gaat
        game.physics.arcade.moveToObject(enemyBullet,player,120);
        // wacht twee seconden voor het volgende schot
        firingTimer = game.time.now + 2000;
    }
}

```

Net zo goed als dat je het jezelf makkelijker kan maken door sneller en meer te vuren kan je het de aliens ook makkelijker maken om jou te raken. Zo wordt er nu maar elke 2 seconden gevuld. En de snelheid van de kogel is 120 pixels per seconde. Die getallen kan je dus veranderen als je meer uitdaging wilt.

Zodra onze kogel een alien raakt moet de alien natuurlijk verdwijnen. Dat gebeurt in de functie **collisionHandler**.

```

function collisionHandler (bullet, alien) {

    // Als een kogel een alien raakt verdwijnen ze allebei
    bullet.kill();
    alien.kill();

    // De score wordt hoger
    score += 20;
    scoreText.text = scoreString + score;

    // Laat een explosie zien
    var explosion = explosions.getFirstExists(false);
    explosion.reset(alien.body.x, alien.body.y);
    explosion.play('kaboom', 30, false, true);

    // Als alle aliens weg zijn heb je gewonnen!
    if (aliens.countLiving() == 0)
    {
        score += 1000;
        scoreText.text = scoreString + score;

        enemyBullets.callAll('kill', this);
        stateText.text = " You Won, \n Click to restart";
        stateText.visible = true;

        //de "click to restart" handler
        game.input.onTap.addOnce(restart, this);
    }
}

```

Zie je dat er een explosie ontstaat als je een alien raakt? Die explosie is een animatie. In de functie **setupInvader** voeg je die toe. Let op: deze functie had je al toegevoegd in je code, dus je hoeft alleen de inhoud van de functie erbij te zetten.

```
function setupInvader (invader) {
  invader.anchor.x = 0.5;
  invader.anchor.y = 0.5;
  invader.animations.add('kaboom');
}
```

Aliens kunnen ook vuren. Als een kogel onze held raakt wordt de functie **enemyHitsPlayer** uitgevoerd.

```
function enemyHitsPlayer (player,bullet) {
  // Haal de kogel weg
  bullet.kill();

  live = lives.getFirstAlive();

  // Haal een leven weg
  if (live)
  {
    live.kill();
  }

  // Laat een explosie zien
  var explosion = explosions.getFirstExists(false);
  explosion.reset(player.body.x, player.body.y);
  explosion.play('kaboom', 30, false, true);

  // Als er geen levens meer zijn is het ... game over.
  if (lives.countLiving() < 1)
  {
    player.kill();
    enemyBullets.callAll('kill');

    stateText.text=" GAME OVER \n Click to restart";
    stateText.visible = true;

    //de "click to restart" handler
    game.input.onTap.addOnce(restart,this);
  }
}
```

Als het spel is afgelopen, omdat je verloren of gewonnen hebt, kan je opnieuw beginnen. De functie **restart** zet alles weer terug.

```
function restart () {  
  
    // zet het aantal levens weer terug  
    lives.callAll('revive');  
  
    // haal de aliens op  
    aliens.removeAll();  
    createAliens();  
  
    // breng onze held weer tot leven  
    player.revive();  
  
    // verberg de tekst  
    stateText.visible = false;  
  
}
```

Het spel is nu klaar om te spelen. Als je tijd over hebt mag je nog geluid toevoegen.

Geluid

Een spelletje wordt natuurlijk leuker als er ook geluid bij zit. In plaats van kant-en-klare stukken code moet je hier zelf wat meer bij nadenken. De geluiden staan in de map **assets**. Je ziet daar

- bullet.wav
- explosion.mp3
- killed.wav
- music.mp3

Om bijvoorbeeld het geluid van een explosie toe te voegen heb je vier regels nodig.

```
var explosionsound;
```

```
game.load.audio('explosion', 'assets/explosion.mp3');
```

```
explosionsound = game.add.audio('explosion');
```

```
explosionsound.play();
```

Bepaal zelf waar je deze regels precies moet zetten. Dus: waar definieer je de variabele, waar laad je het geluid in het geheugen, waar voeg je het geluid toe aan het spel en op welke plaats laat je het geluid horen?

Succes!