Deep Sea Diver

Document:	Implementatie	Opdrachtgever:	Jan Johannes Veldkamp	Telefoon:	+31615295732
Datum:	13-4-2016	Programmeur:	Nick van Urk	E-mail:	nickvanurk@hotmail.com

Inhoudsopgave

1. Inleiding

In dit document wordt mijn aanpak beschreven hoe fouten in mijn code worden opgelost. Tevens wordt de realisatie periode van de applicatie in kaart gebracht om zo te kunnen bekijken wat er nou goed en minder goed ging. Tot slot zal ik uitleggen hoe ik bepaalde interessante spelelementen heb geïmplementeerd.

2. Realisatie

Voordat ik begin met programmeren zorg ik er eerst voor dat ik een goed beeld heb van wat er moet worden gerealiseerd. Een groot deel van het spel moet van te voren daarom al duidelijk zijn. Een spel hoeft niet tot in de kleinste details van te voren worden uitgewerkt, sommige dingen moet je simpelweg testen in de praktijk. Het idee wordt in kaart gebracht door een ontwerp document te schrijven. Dit ontwerp document dient als referentie materiaal gedurende de realisatie periode.

Het ontwerp document wordt gebruikt om het programma in grote lijnen te kunnen uitstippelen. Tijdens deze periode wordt er zorgvuldig nagedacht over hoe allerlei ideeën daadwerkelijk kunnen worden geïmplementeerd. Een programma zoals een computerspel valt natuurlijk op te splitsen in veel kleinere concepten. Tijdens het bouwen maak in gebruik van een methode genaamd 'bottom-up design' dit wil zeggen dat ik eerst een module van het spel ontwikkel om deze hierna vervolgens uit te breiden met nog een module. Het programma begint klein maar naarmate de tijd vordert groeit deze in complexiteit en volledigheid.

Tijdens het bouwen van de applicatie wordt er gedebuged bij elke verandering in de code die plaatsvind. Dit wil zeggen dat er wordt gekeken of het programma fouten bevat en zo ja waar deze fouten zich bevinden zodat de programmeur (ik) de deze kan herstellen. Als er geen fouten zijn gevonden zal het programma worden gebouwd waarna ik deze zal uitvoeren. Hierna kijk ik of de aanpassing werkt zoals ik het voor ogen heb. Tijdens het testen probeer je eigenlijk altijd de code te breken. Zo kom je achter je logica fouten, deze fouten worden niet gedetecteerd door de debugger omdat hier de syntaxis niet incorrect is maar de logische werking ervan. De applicatie zal hierdoor onvoorspelbaar gedrag vertonen. Door frequent te testen verklein je de kans op fouten en maakt het je programma betrouwbaarder.

Als de applicatie is voltooid zal ik deze grondig testen. Dit houdt in dat ik de applicatie in het geheel controleer op fouten en probeer te breken, ik zal zo veel mogelijk handeling uitproberen en zo de applicatie op alle mogelijkheden testen. Ook wordt de code onder de loop genomen om te kijken wat hier nog valt te verbeteren en eventuele fouten op te lossen. Als ik helemaal tevreden ben

over de applicatie pas dan is deze voltooid en wordt de bijbehorende documentatie verder uitgebreid.

2.1 Verloop

In totaal ben ik vijf weken bezig geweest met het realiseren van de applicatie. De enige afspraak die is gemaakt is dat ik minimaal één keer per week met de opdrachtgever in gesprek ga over de huidige status van het project. In plaats om alles in één lap tekst op te sommen geef ik hieronder per week een gedetailleerd overzicht. Ik ga in op wat er is gerealiseerd maar ook wat er goed ging en waar ik tegen aan ben gelopen tijdens de realisatie.

2.1.1 Week 1

Als eerst heb ik alle benodigde programma's en libraries geinstalleerd op mijn laptop zodat ik me daar niet meer mee bezig hoefde te houden en mij volledig kon gaan focussen op het realiseren van de applicatie. Omdat het hier om een computerspel gaat dat web technologieën gebruikt heb ik in de eerste week eerst mijn kennis aardig moeten bijspijkeren. Web is niet mijn specialiteit. Om deze reden heb ik veel informatie moeten opzoeken en lezen. De realisatie kwam hierdoor moeilijk op gang. Aan het einde van de eerste week had ik nog niet iets noemenswaardig gerealiseerd maar al wel veel van de benodigde kennis opgedaan. De basis. Het tekenen van een plaatje op het beeldscherm, beweging, de input van het toetsenbord etc...

2.1.2 Week 2

Week twee was niet zo heel veel anders dan de eerste week. Ik had nog steeds veel moeite en er leek maar geen gang in te komen. Omdat ik de basis al een beetje in de vingers had heb ik mij echter meer kunnen focussen op hoe ik bepaalde spel elementen zou kunnen realiseren. Daar heb ik vooral in deze week mee geexperimenteerd. Aan het einde van deze week had ik echter wel al iets om te laten zien aan de opdrachtgever al was het niet veel. Het genereren van het level was nu geïmplementeerd. De opdrachtgever zag zijn ideeën tot leven komen en reageerde enthousiast wat mij weer meer motivatie gaf. Ook heb ik mijzelf constant kunnen verbazen door dingen te realiseren waar ik voorheen nog helemaal geen enkele ervaring mee heb gehad. Zowel technologieën als achterliggende concepten.

2.1.3 Week 3

Week drie was toch echt wel dé week. Op dit punt tijdens de realisatie periode had ik genoeg kennis opgedaan van de gebruikte technologieën op vlot dingen te kunnen realiseren en alles leek gemakkelijker te gaan. In deze week heb ik bijna alle kern spelelementen kunnen implementeren: het le-

vel, de speler, beweging, gebruiker input en collision detection. De applicatie begon al echt op een spel te lijken. Aan het einde van deze week heb ik weer met de opdrachtgever rond de tafel kunnen zitten en deze was ook zeer enthousiast over het resultaat wat mij weer deugd deed. Ook heb ik tijdens deze week mijn oude code kunnen opschonen omdat ik meer kennis had opgedaan dus nu veel ruimte zag voor verbetering met name hoe het level wordt gegenereerd.

2.1.4 Week 4

In week vier heb ik meer aandacht kunnen besteden aan de statistieken van het spel. Ik heb met name de levens van de speller geïmplementeerd met bijbehoren 'health bar' en tevens de score / dieptemeter. Ook heb ik het spel een kleine upgrade gegeven wat het uiterlijk betreft. De tunnel zag er nu ook daadwerkelijk uit als een tunnel in plaats van vierkantje blokken. Aan het einde van de week heb ik mij ook kunnen verdiepen in game staten denk hierbij aan een game-over scherm of een hoofdmenu. De eerste heb ik kunnen implementeren en laat netjes de behaalde en hoogst behaalde score van de speler zien. De opdrachtgever en ik waren wederom tevreden over het nu al behaalde resultaat. Nog elke dag leer ik er dingen bij en probeer ik oude code te verbeteren en nieuwe elementen aan het spel toe te voegen alhoewel nu alles er toch al bijna in zat. Ook heb ik research gedaan naar andere soortgelijke spellen en enkele mock-ups in Photoshop gemaakt qua user-interface en game staten betreft.

2.1.5 Week 5

In de laatste week van de realisatie heb ik de overige game-staten geïmplementeerd zoals het titel het how-to scherm. Ook zitten er vanaf deze week geluid in het spel en daarmee is het spel toch echt tot leven gekomen. Verder heb ik het uiterlijk van het spel nog wat kunnen verbeteren door oude art van het spel te vervangen. Aan het einde van deze week was ik zeer tevreden met het behaalde resultaat. Na wederom een gesprek te hebben gehad met de opdrachtgever zijn we het er beide over eens geworden dat het spel af is en tevens voldoet aan het ontwerp van het spel dat wordt beschreven in het ontwerp document.

2.2 Level Generatie

In het spel 'Deep Sea Diver' is het meest interessante het random gegenereerde level, dit is ook gelijk de meest interessante code. Om deze redenen ga ik beschrijven hoe ik dit heb gerealiseerd. Van concept tot implementatie.

De tunnel moest aan een paar eisen voldoen:

- De tunnel moet in het midden beginnen.

- De tunnel gaat willekeurig naar links of rechts.
- De tunnel kan niet verder dan de rand van het scherm.
- De tunnel moest er grafisch mooi uitzien door middel van textures.

Om de tunnel er grafisch mooi uit te laten zien heb ik besloten om de data van de tunnel eerst te genereren om hier vervolgens een tunnel rij mee te creëren. Hierdoor kan ik de randen van de tunnel rij een correcte facing geven.

Hieronder volgt de code van mijn implementatie voor het genereren van level data. Deze functie wordt aangeroepen in een andere functie voor het creëren van een tunnel rij.

```
function generateDataRow() {
      // Create the initial data row
      var row_data = {
            left_width: getRandomNumberBetween(1, 3),
            right_width: getRandomNumberBetween(1, 3)
      };

      // if the level data array is NOT empty
      if (level_data.length) {
            var last_row = level_data[level_data.length - 1];

            // for every new data row, increment its width.
            row_data.left_width += last_row.left_width;
            row_data.right_width += last_row.right_width;

            var gap = level_width - row_data.left_width -
                  row_data.right_width;

            // Start generating the tunnel
            if (gap < tunnel_width) {
                  // Make the tunnel go left
                  if (Math.random() < 0.5) {
                     var width = last_row.left_width - getRandomNum
                        berBetween(1, 3);

                     row_data.left_width = width <= 0 ? 1 : width;
                     row_data.right_width = level_width -
                        (row_data.left_width + tunnel_width);
                  }
                  // Make the tunnel go right
                  else {
                     var width = last_row.right_width - this.getRandomNum
                        Between(1, 3);

                     row_data.right_width = width <= 0 ? 1 : width;
                     row_data.left_width = level_width -
                        (row_data.right_width + tunnel_width);
                  }
            }
```

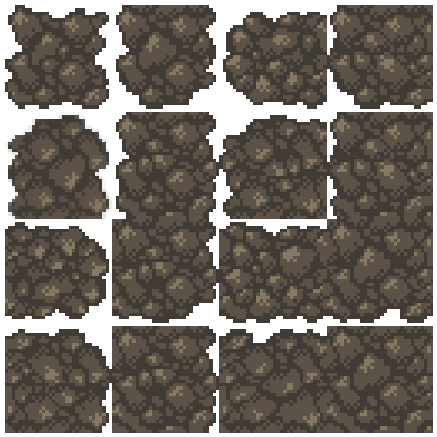
```
            // We only need to store the last three data rows in order to
            // generate a tile row with the correct facing.
            if (level_data.length > 3)
                  level_data.shift();
      }

      // Push the data row into the level data array
      leve_data.push(row_data);

      // Return the data row for potential further calculations
      return row_data;
};
```

In de volgende functie wordt er een tunnel rij gecreëerd. Deze functie wordt aangeroepen in de initialisatie van het spel om de eerste rij te creëren. Op deze rij wordt vervolgens verdere logica uitgevoerd zoals het bewegen of vernietigen ervan.

```
function createTileRow(posY) {
      if (!level_data.length)
            generateDataRow();

      var previous_data_row = level_data[level_data.length - 2];
      var current_data_row  = level_data[level_data.length - 1];
      var last_data_row     = generateDataRow();

      for (var i = 0; i < level_width; ++i) {
            // Only check the tiles, skip the gap
            if (i < current_data_row.left_width ||
                i >= level_width - current_data_row.right_width) {
                  var top = bottom = left = right = 0;

                  // Is there a tile above this one?
                  if (previous_data_row) {
                        if (i < previous_data_row.left_width ||
                            i >= level_width - previous_data_row.right_width) {
                              top = 1;
                        }
                  }

                  // Is there a tile below this one?
                  if (i < last_data_row.left_width ||
                      i >= level_width - last_data_row.right_width) {
                        bottom = 4;
                  }

                  // Is there a tile left of this one?
                  if (i <= current_data_row.left_width ||
                      i > level_width - current_data_row.right_width) {
                        left = 8;
                  }

                  // Is there a tile to the right of this one?
                  if (i < current_data_row.left_width - 1 ||
                      i >= level_width - current_data_row.right_width - 1) {
                        right = 2;
```

```
                }

                // Create the tile with the correct facing and position
                var tile = new Phaser.Sprite(game, 0, posY, 'rock_4x4');
                tile.x = i * TILE_SIZE;
                tile.frame = top + bottom + left + right;
                game.physics.arcade.enable(tile);
                tile.body.immovable = true;
                tiles.push(tile);
                game.world.addChild(tile, 0);
            }
        }
};
```

De facing van een tile wordt bepaald door bitwise methode. Elke tile krijgt een nummer afhankelijk van zijn solide 'buren' die ook een nummer hebben. Dit wordt bij elkaar opgesteld en de uitkomst wordt gebruikt om de juiste frame uit een texture op te halen. In de onderstaande afbeelding representeert elk vierkant een frame dus van 0 t/m 15.



Vervolgens zal ik alle level tiles (opgeslagen in de array 'tiles') omhoog laten bewegen met de volgende functie die wordt aangeroepen in de update functie van het spel. Deze update functie wordt 60 keer per seconde aangeroepen door het programma.

```
function moveTilesWithSpeed(scroll_speed) { // scroll_speed = 64 px per second
      // Get the last tile in the array
      var i = tiles.length - 1;

      while (i >= 0) {
            var tile = this.tiles[i];

            tile.body.velocity.y = -scroll_speed;

            // If the tile is off-screen, destroy it
            if (tile.y + TILE_SIZE < 0) {
                  tiles.splice(i, 1);
                  tile.destroy();
            }

            i--;
```

```
	}
};
```

Hierna wordt er in de update functie nog een stuk code uitgevoerd om te bepalen wanneer er een nieuwe level (tile) rij moet worden gecreëerd.

```
var last_tile_y = this.tiles[this.tiles.length - 1].y;
if (last_tile_y <= game.world.height) {
	createTileRow(last_tile_y + TILE_SIZE); // y-axis
}
```

2.3 Score

Hier zal ik beschrijven hoe ik de score van het spel heb geïmplementeerd. De score loopt geleidelijk op maar als de speler zich naar boven beweegt zal dit langzamer gebeuren en als de speler zich omlaag beweegt zal dit sneller gebeuren. De reden hiervoor is omdat de score een diepte moet voorstellen. De code wordt wederom in de update functie van het spel uitgevoerd.

```
var interval = Math.abs(1 / ((scroll_speed + player.body.velocity.y) / TILE_SZE)
* 1000);
```

De interval variabel staat voor een tijd in milliseconden. Als eerste deel ik 1 door een getal. 1 staat voor 1 seconde en het getal staat voor een snelheid per seconde. Deze snelheid is afhankelijk van de snelheid van de speler. Als de speler zich omhoog beweegt zal de snelheid van de speler een getal onder de 0 zijn en als de speler zich omlaag beweegt zal dit een getal boven de 0 zijn. Als de speler zich niet beweegd zal de snelheid van de speler 0 zijn. De scroll_speed is 64 pixels per seconde en de TILE_SIZE is 32 pixels. Wat je hieruit kunt opmaken is dat als de speler zich niet zou bewegen (snelheid is 0) dan zou de interval een halve seconde zijn oftewel 500 milliseconden. Omdat de tiles in het level met dezelfde scroll_speed snelheid wordt verplaats staat deze interval tijd ook gelijk aan de tijd dat het duurt voordat een tile zich van zijn huidige positie TILE_SIZE(32) pixels heeft verplaats (als de snelheid van de speler gelijk is aan 0.) Er is dus een relatie tussen het level en de speler. Op deze manier kan ik een diepte meter simuleren.

Voorbeelden:

```
1 / ((64 + 0)   / 32) * 1000 = 500  milliseconden
1 / ((64 + -48) / 32) * 1000 = 2000 milliseconden
1 / ((64 + 48)  / 32) * 1000 = 285  milliseconden
```

Vervolgens de rest van de score code:

```
if (getTimerAsMilliseconds() > interval) {
	if (scroll_speed + player.body.velocity.y > 0)
		score++;
```

```
	else if (scroll_speed + player.body.velocity.y < 0 && score)
		score--;

	resetTimer();
}
```

3 Conclusie

Ik ben zeer tevreden over het verloop van de algehele realisatie periode alhoewel het begin vrij stroef verliep. Af en toe zat ik even vast met een probleem maar door goed na te denken of simpelweg op het internet te kijken kwam ik al snel met een oplossing. Ik heb veel kennis op gedaan qua web technologieën zowel als achterliggende spel concepten. De opdrachtgever was zeer te spreken over het eindresultaat en is hier dan ook mee akkoord gegaan. Persoonlijk vond ik het een leuk project om te verwezenlijken. Qua deadline betreft is dat ook helemaal goed gekomen. Voordat ik begon had ik voor mezelf 6 weken ingepland voor de realisatie waarvan ik er maar 5 nodig heb gehad.