**PWS middagen op school:**

Tijdens de PWS middagen hebben William en ik de game wat meer uitgedacht: Wat voor spel wordt het, welk doel, welke functies heeft het spel etc. Daarna hebben we taken verdeeld. Ik ga me vooral focussen op de code. William gaat zich richten op de muziek, graphics, verhaal en alles dat er nog meer bij komt kijken om een game te maken.

**18 September:**

Oriëntatie op het onderwerp: Hoe zit een Java 2D game in elkaar? YouTube filmpjes opgezocht waarin beschreven werd hoe Graphics2D werkt. Ongeveer 2 uur bezig. Ook heb ik geleerd hoe je een jar file uitvoerbaar maakt.

**19 September:**

Heb geleerd hoe Threads in Java werken. Deze ga ik gebruiken om de game te laten loopen.

**23 September:**

Proberen een eigen programma te schrijven in Java met Graphics2D. Na zo’n 4 uur is het me gelukt om een programma te maken dat een scherm opent dat een willekeurige kleur genereert om de seconde. Nu weet ik hoe JFrame werkt en hoe ik een JPanel hierin moet renderen.

**24 September:**

Proberen toetsaanslagen te registeren in het voorgenoemde programma. Na een uur of 2 is het gelukt om een pop-up te generen in het scherm wanneer een toets wordt ingedrukt.

**1 October:**

Op school 2 tussenuren achter elkaar gewerkt aan het PWS. Vooral op YouTube filmpjes gekeken waarin mensen platformers en andere applicaties maken in Graphics2D.

**3 October:**

2 tussenuren achter elkaar source code opgezocht van games die geschreven zijn in Java via Grapgics2D. Het ziet er best gecompliceerd uit.

**14 October:**

Source code van een simpele space invaders game bekeken op youtube. Ziet er allemaal best begrijpelijk uit.

**20 October:**

3 tussenuren source code bekeken van games in Java Graphics2D. Een hiervan is een project op youtube. Deze gebruikt ook een soort level manager dat we dus al van plan waren. Dan werkt het dus wel.

**21 October:**

Begonnen aan het project! In IntelliJ heb ik een nieuw project aangemaakt dat nu “PWS” heet. De naam komt later wel. Tot nu toe is het project alleen nog maar een Main class en een class die JPanel implementeert.

**3 December:**

Na een behoorlijke “pauze” waarin in eigenlijk alleen maar bezig ben geweest met school en uren per dag programmeren aan mods voor mijn game community, ben ik serieus begonnen met de grotere lijnen van het project.

Om de een of andere reden loopt de game niet. Dit heb ik opgelost door een paar methods weg te halen en weer neer te zetten. De echte fout heb ik niet gevonden maar spontaan werkte het weer.

Hierna heb ik een LevelManager class, Level interface en Achtergrond class toegevoegt. De levelmanager zal steeds de methods doorgeven aan het gewenste level. De achtergrond zal steeds loopen om beweging te simuleren.

Daarnaast heb ik diep nagedacht hoe de structuur van het project zal worden en hier heb ik lege classes voor aangemaakt.

Hier heb ik toch voornamelijk de dag mee gevuld. Ongeveer 8-9 uur.

**4 December:**

Om het makkelijker te maken om overal aan het PWS te werken heb ik het project op Github gezet. Ook bouw ik alles nu in Maven omdat het gewoon makkelijker is.

Daarnaast heb ik zeker weer zo’n 8 uur lang classes uitgebreid. Voornamelijk speler locatie, speler renderen en achtergronden goed laten loopen.

Ik heb ook een paar util classes gemaakt die van een PGN een animatie maken. Er is nu ook een centrale class die objecten in de wereld rendert relatief aan de speler. Dit is erg belangrijk omdat de speler eigenlijk stil staan in het scherm. Alleen de y coordinaat verandert. De x wordt groter maar hij rendert de speler altijd op een vaste x positie. Alle andere objecten bewegen om de speler heen. De locatie wordt steeds door een formule berekent.

**5 December:**

Objecten! En dan echte die goed bewegen. Er is één object class met een constructor die veel parameters accepteert zodat je het object kan veranderen.

**6 December:**

We hebben nog geen goede naam. William stelde voor: “Super Duper Platformer Simulator 2014”.

Daarnaast heb ik objecten nog aangepast. Objecten die bewegen in een radius (Heen en weer om een vaste coordinaat) flippen nu als ze achteruit lopen. Zo is er dus maar 1 image nodig. Best handig!

Coordinaten heb ik nu ook opnieuw gedaan. Er is nu een Location class die een getX(), getY() en getRenderedY() methode bevat. Zo heb ik dus altijd een compact pakket informatie over de locatie ten opzichte van de speler. getRenderenY() geeft de hoogte van het scherm – getY(). Y = 0 zou normaal boven in het scherm zijn. Nu is dat dus onder. Logischer als je er over na denkt.

**7 December:**

Niet heel veel er aan gewerkt. Wel heb ik ontdekt dat ik een willekeurig pictogram kan gebruiken voor het scherm. Dit heb ik dus toegevoegt. Het ziet er nu toch iets meer customized uit.

Verder nog zo’n anderhalf uur optimalisatie gedaan. Objecten die op een verkeerde plek stonden meer centraal neergezet. Kort gezegt een logischere structuur.

**8 December:**

Objecten renderen, maar de graphics zijn scheef. Ik heb in de centrale render class een aanpassing gedaan waardoor de methode het centrum van de afbeelding bepaald. Zo zal de locatie waarop je een object laad het centrum van het plaatje zijn. Dit is heel handig voor de hitboxes die later nog komen voor collision.

**8 + 9 December:**

Ongeveer 5 uur tot 3 uur ’s nachts geprogrammeerd. Ik heb nagedacht en kwam tot de conclusie dat ik heel veel redundand code had. De objecten in de wereld zouden veel beter een vaste structuur kunnen hebben en dan abstract zijn. Dit heb ik dus gedaan. Na een tijdje dingen uitproberen is het gelukt en het ziet er nu echt een stuk beter uit. Verder is het zo beter geoptimaliseerd want de objecten zijn nu kleiner. Ook kost het nu veel minder tijd om objecten in de wereld te laden.

Daarnaast heb ik de grondstukken gelegd voor collision. Er is nu een parameter voor hitbox grootte. Deze hoeft niet ingevuld te zijn en zal, wanneer niet ingevuld, even groot als de image van een object zijn.

Ook heb ik nog “damage” toegevoegt aan objecten. Dit betekent dus de damage die ze aan de speler zullen doen zodra ik dat toe ga voegen.

**9 December:**

Images zagen er niet goed uit: Ze waren heel hoog en niet breed. Schaal van 1:60 ongeveer. Dit moest dus wel te maken hebben met de class die een image in verschillende frames verdeeld. Het bleek dus dat die class niet de afstand tussen de frames als parameter kreeg, maar de schaal van de images. Dit zorgde dus voor het probleem. Na kort zoeken vond ik dat scale en distancebetweenframes inderdaad omgedraaid waren in een van de classes.

Vectoren toegevoegt. Hebben een snelheid en richtingscoefficient. Deze kan ik gebruiken om projectielen te lanceren.

Verschillende soorten objecten defined. Stilstaande, volgende en lopende wezend. Verder projectielen, silstaande objecten en bewegende objecten. Ze animeren allemaal volgens de super class, alleen de bewegende objecten updaten hun locatie nog. Het “StalkingCreature” object zal zijn vector steeds aanpassen om dichter bij zijn target te komen.

Grote verandering gemaakt: eerst had alles een vaste snelheid. Deze was of 0 of die snelheid. Nu werkt het anders. Er is nu een update methode voor alles dat beweegt. Dit zal de objecten met hun snelheid laten bewegen. Verder is er nu een velocity class. Deze zal de snelheid automatisch laten dalen. De snelheid zal dus beinvloed kunnen worden door meerdere factoren. Dit was nodig om zwaartekracht te kunnen maken op een later moment.

De speler class die eerst alleen locatie regelde doe ik nu ook helemaal opnieuw. Ik heb een gedeelte van deze class al gemaakt. Het bevat nu alle informatie over de speler: Snelheid, locatie en levens. Dit zal later meer informatie worden naar mate er meer opties in het spel komen. Bijvoorbeeld een cooldown op damage geven en krijgen. En hoeveel damage een speler doet.

**10 December:**

Vroeg in de ochtend op laptop gegaan om nog even wat enums af te maken. Deze enums zijn er voor elk soort object. Zo kan ik makkelijk nieuwe objecten maken met een <enum>.<object>.build(x, y) op een bepaald coordinaat.

**13 December:**

Na een paar dagen pauze heb ik pogingen gedaan om een oplossing te vinden om alle objecten tegelijk te laden. Het probleem is dat ze allemaal anders zijn. De oplossing is eigenlijk best simpel: Ik moet het simpelste object nemen (de Object class in dit geval) en die in een map opslaan. Daarna kan ik kijken of ze een instance zijn van een bepaalde class. Als dat zo is kan ik ze makkelijk casten en de methodes uitvoeren.

Het volgende probleem is hoe ik de coordinaten van objecten ga bijhouden. Ze bewegen namelijk ook. Alle objecten altijd af gaan en kijken of ze geladen moeten worden zou heel veel werk zijn. Ik denk dat ik dit ga oplossen door twee maps te maken. Een zal alle objecten bevatten en langzaam daar doorheen gaan en steeds de relevante objecten in een tweede map stoppen die steeds geladen wordt. Zo wordt er niet steeds door alle objecten gecycled.

**16 December:**

Een paar dagen druk bezig geweest met het voorbereiden van mijn Nederlands betoog. Vandaag heb ik een paar redundant methods weggehaald. Hitboxradius klopt nu ook beter want eerst nam het het HELE plaatje als breedtje, niet alleen het gerenderde frame. Daarnaast heb ik projectielen nog een check gegeven waardoor ze zichzelf verwijderen zodra ze uit het zicht zijn.

Ook heb ik nagedacht over hoe de vloer, botsen en zwaartekracht het best kunnen werken. Ik ben op het volgende idee gekomen:

Als vloer wilde ik eerst overal los een stukje vloer neerzetten. Dit is echter best wel intensief. Het lijkt me dus beter om, net als bij achtergronden, een aparte class te nemen en deze voor elk level te laten verschillen. In deze class kun je dus een lijst stoppen van plekken waar GEEN vloer is. Zo is de vloer niet een berg losse objecten, maar meer 1 object.

Voor de zwaartekracht voeg ik gewoon een check toe die aan alle objecten die een instatie zijn van LivingObject de velocity Y omlaag doet. Dus steeds -1 per seconde ofzo. Dan krijg je dus het effect dat je steeds sneller gaat vallen. Wel zal ik kijken of mensen botsen voor dat dit wordt toegevoegt. Zodra iemand de grond of een object raakt zal het weer op 0 worden gezet.

Voor het botsen had ik al eerder rectangles toegevoegt (die ik nu heb verbeterd). Ik ga dus een class maken waarin een check alle objecten die geladen zijn met elkaar gaat vergelijken. Als ze elkaar raken zal dit tellen als een botsing. Alle objecten hebben een “Action on collision” en “Collides” parameter. Deze zegt dus: Kan je er doorheen lopen of niet, en wat gebeurt er als je dit object raakt.

Zo zal ik dus steeds de spelen locatie verandering tegenhouden als hij botst. Dit ga ik proberen ook voor alle objecten in de wereld te doen om het een dynamisch geheel te maken. Ik hoop wel dat dit niet te intensief wordt voor het spel aangezien het dus zo’n 60 keer per seconde door alle objecten zou moeten loopen.

Later vandaag:

Ik ben begonnen met het maken van de vloer. Dit doe ik met de nieuwe Tile en Floor class. Elke level class implementeert dit. Om de een of andere reden worden tiles wel verkeerd geladen. Dit moet wel te doen hebben met de methode waarbij ik bereken waar een tile moet zijn. Dit doe ik met %.

Edit: Fout gevonden! Ik vergat de objecten in de wereld te renderen en in de Tile class berekende hij de laatste tile die geladen moest worden verkeerd. Hij nam de uiterste coordinaat die zichtbaar was en deelde die door de breedte van 1 tile. Wat overbleef nam hij. In plaats van een % moet ik hier gewoon een / neerzetten.