

sLIME vOLLEY

Programmeren



**GILLES BALLEGEER & GARBEN TANGHe**

21/05/2017

UGENT

Belangrijkste ontwerpsbeslissingen

Om het coördinatensysteem te vereenvoudigen hebben we een COY(y) functie aangemaakt.

Deze zet een coördinaat y (uit het originele coördinatensysteem) om in een nieuwe y-waarde met z’n coördinatensysteem als volgt:

* De x-waarden blijven onaangepast.
* De y-waarden starten op 300 px en lopen naar boven toe i.p.v naar beneden.

Dit maakt het voor ons eenvoudiger om te rekenen met coördinaten.

Iets over AI level 3?

Om het spel eerlijk te maken, laten we telkens de verliezer opslaan.

In de code maken we gebruik van een for each lus i.p.v. een iterator uit de STL library omdat dat sneller te implementeren is en ook overzichtelijker is. Dit kan echter wel gepaard gaan met een lichte achteruitgang van de snelheid, maar dit zorgt niet voor merkbare problemen.

Wanneer de speler op ESC drukt tijdens een spel, hebben we ervoor geopteerd om het spel niet op te slaan. De speler heeft namelijk het spel niet uitgespeeld.

Om de samenwerking te vergemakkelijken hebben we ervoor geopteerd om GitHub te gebruiken. Zo hadden we geen problemen met overlappende versies van bestanden.

Taakverdeling

Gilles: voornamelijk focus op single player game en replay.

Garben: voornamelijk focus op multiplayer game. De geheugenlekken heb ik ook op me genomen.

We hanteerden elk een totaal andere methode van programmeren. Gilles begon onmiddellijk met programmeren en schreef dan ook de meeste code, terwijl Garben eerder afwachtte tot hij precies wist waar de code moest staan en wat het moest doen. Op die manier vond hij dan bugs en kon hij de code van Gilles copy-pasten in het juiste systeem. Desondanks onze verschillende werkwijze konden we toch goed samen werken en vulden we elkaar goed aan. Diversiteit in onze groep was in dit geval geen nadeel, integendeel.

Problemen bij uitvoering

We hebben vaak gesukkeld met de offsets op de sprites. Het heeft soms toch wel even geduurd vooraleer we de tekeningen op de juiste plaats verkregen.

We hadden in het begin niet onmiddellijk door waarvoor elk van de systemen gebruikt moest worden. We gebruikten system\_input\_single, system\_input\_multi, system\_state\_single & system state\_multi eerst niet en toch werkte het spel. We hadden namelijk de werking van de systemen die input zouden moeten behandelen geimplementeerd in system\_motion zelf. Dit zorgde er dan weer voor dat we wat meer code nodig hadden om te controleren of we in single- of in multiplayer aan het spelen waren. Na heel wat copy-pasten van de code op de correcte plaats, verkregen we uiteindelijk toch de gebruikelijke werking van het spel.

Wat we momenteel nog niet goed begrijpen is hoe de systemen aan de game engine worden toegevoegd. We hebben het nu gedaan door een pointer naar het systeem (een variabele van de klasse) mee te geven, maar we vonden nergens terug waar de systemen precies worden aangemaakt.

Bugs of TODO’s:

De bal reageert anders in een single player game dan in een multi player game.

Highscore en replay moeten nog gedaan worden.

We kunnen vaker gebruik maken van x\_min, x\_max, x\_of, … bij sprite components i.p.v. telkens met getallen te werken.

AI level 3

Collision tussen speler en net of muur