Space invaders

Design

Het spel

De speler bedient een schip via thrusters. Het simuleert dus een beetje de vacuum. Het is de bedoeling de vijandige schepen neer te schieten. Je mag ook zelf niet geraakt worden.

Ook de randen aanraken is een lose condition. Wanneer alle vijanden dood zijn ga je automatisch naar het volgende level.

Mapstructuur

Het eerste level is onderverdeeld in een map voor de binaire files en voor de source, zodat builds gescheiden zijn van de sourcecode. De sourcemap is onderverdeeld in cpp- en headerfiles, zo is er een duidelijk verschil tussen de implementaties en de specificaties. Ook voorkomt dit dat er enorm veel files in 1 map zijn.

Het bestand met de main functie word gewoon in de sourcemap gezet om duidelijk te maken waar het programma begint.

Model

De game heeft een gameloop verdeeld in frames met vaste deltatijd. Dit om er voor te zorgen dat de physics deterministisch verlopen. Het game model heeft een lijst met pointers naar de entiteiten die zich in de wereld bevinden. De game is verantwoordelijk om de entiteiten te laten weten wanneer ze botsen met andere objecten alsook om hun vaste update te callen.

View

De game wordt zolang het scherm open is zo snel mogelijk getekend. Het is echter makkelijk de framerate te limiteren in sfml.

De objecten zijn zelf verantwoordelijk om te zichzelf te kunnen tekenen.
Ze krijgen daarvoor een window waarop ze zich moeten tekenen. De view laat de entiteiten weten wanneer ze zichzelf moeten tekenen.

De view wordt hertekend wanneer het model via het observer pattern de view notified.

Controller

Controllers hebben een pointer naar de entiteiten die ze beheren. Bijvoorbeeld de playercontroller kan de thrusters van de speler beheren.

DeltaTime

Om de physics deterministisch te simuleren hebben we een vaste timestep nodig. Om dit te bereiken zijn de physic ticks en de fps ticks niet aan elkaar gebonden. De frames worden getekend aan de maximale snelheid. De frames worden echter maar berekend met 5 milliseconden intervallen. Als de computer niet sterk genoeg is om dit te halen kan de deltaTime verhoogd worden.

Een deterministische simulatie is makkelijker te debuggen en geeft mijn inziens een elegantere code. Echter zonder interpolatie bij de view wilt dit ook zeggen dat de fps gelockd is. Wanneer de deltaTime groter is dan de tijd tussen 2 mogelijke tekenframes zou er van interpolatie gebruik kunnen gemaakt worden. Dit lost ook het probleem op waarbij te weinig rekenkracht de deltaTime enorm doet toenemen. Aangezien interpolatie waarschijnlijk een goedkopere actie is dan de physics te berekenen.

Op deze manier is het echter niet nodig om de view te notificeren als het model veranderd. Omdat de view zo snel mogelijk hertekend terwijl de model fixed is.

Community input:
http://gamedev.stackexchange.com/questions/1589/when-should-i-use-a-fixed-or-variable-time-step

Botsing

Botsing is simpel. Elk entity type kan een collision functie maken dat wordt gecalled met het specifiekste type dat werd geïmplementeerd. Dit is mogelijk dankzij double dispatch.

https://en.wikipedia.org/wiki/Double_dispatch

Exceptions

Errors worden via exceptions verwerkt. Er zijn verschillende type exceptions die allemaal erven van andere exceptions. Zo is het perfect mogelijk om goede try catch blokken te schrijven voor de verschillende soorten errors.

Cmake

Als build systeem wordt er cmake gebruikt. Het heeft een handige interface en compiled alles in stukken waardoor het vaak een pak sneller gaat.