Documentatie Asteroids v1.0

Gemaakt door: Emiel de Brouwer & Robbert Goey

Gebruiksaanwijzing Asteroids

Je speelt in een Third Person View. Gebruik de WASD-toetsen op het toetsenbord om jezelf te bewegen: druk op "W" om vooruit te lopen, druk op "A" om links of "D" om rechts te keren en druk op "S" om achteruit te lopen. Om te schieten druk je op de "Spacebar". Zorg dat je niet geraakt wordt door een asteroide, want je hebt maar 1 leven! Om te winnen schiet je alle asteroiden kapot! Je kunt eventueel de asteroiden aanpassen (meer, minder, sneller, langzamer, etc.) in de level.txt. Om informatie over de asteroide op te vragen klik je met je muis op een asteroide.

HUD

De HUD laat de volgende teksten zien:

- De tijd
    - We gebruiken de tijd op dezelfde manier als hoe we dat gedaan hebben met de Computer Graphics taak met behulp van SFML. De tijd loopt meteen zodra het spel is opgestart. De tijd wordt ook gebruikt als score. Hoe minder lang je erover hebt gedaan hoe beter je score is.
- Asteroide informatie
    - Door middel van picking, kan er op een asteroide geklikt worden met de muis, waardoor er informatie op het scherm wordt getoond, in dit geval de huidige locatie van de asteroide. De informatie gaat weg als de speler op enter drukt.
- FPS
    - Voor debug mogelijkheden, kunnen we de Frames Per Second tonen, niet te verwarren met First Person Shooter ;)
- Huidige locatie van de speler
    - Voor debug mogelijkheden tonen we de locatie van de speler.

Requirements

De volgende requirements zijn in de game geïmplementeerd:

- Ruimteschip vliegt door een 3D ruimte met asteroïden
- Asteroïden hebben elastische botsingen met elkaar en "muren"
- Asteroïden worden verwijderd zonder memory leak
- De camera drift soepel achter het ruimteschip (met behulp van behaviour)
- Een octree
- Broad phase controle op botsingen
- Kapot schieten van asteroïden
- Picking
- Bepalen welke triangle van het ruimteschip wordt geraakt
- Narrow Phase Collision Detection

Gebruikte patterns

Visitor Pattern

We gebuiken het visitor pattern voor de renderer, de updater en de collisions. Dit zorgt ervoor dat alle render, update en collision operaties bij elkaar in een klasse zitten ipv verspreid over meerdere klasses. Ook een voordeel is mochten we een nieuwe visitor toe willen voegen dan hoeven hiervoor geen bestaande klasses worden aangepast.

We hebben ervoor gekozen om het objectmodel de traversal te laten doen in plaats van de visitor.

Het objectmodel zorgt ervoor dat kinderen ook gevisit worden, dus ook de accept methode van alle kinderen aanroepen. Ook de mesh, behaviour en collider van een object vallen hieronder.

Rendervisitor

Alle draw methoden uit de Visitable klasses worden vervangen door de visit methodes van de RenderVisitor. Behaviour en Collider hoeven niet te worden gerenderd dus die visit methodes doen niets.

Builder en factory

Voor de opbouw van de levels gebruiken wij het Builder en factory pattern. De builder leest een textfile uit (“level.txt”, standaard). Vervolgens gebruikt deze builder de factory om alle objecten aan te maken.

Singleton

Voor de tijd gebruiken we de Singleton Pattern. Hiermee zorgen we ervoor dat de toegang tot de tijd altijd via één object te laten gaan. Hierdoor is de tijd uniek.

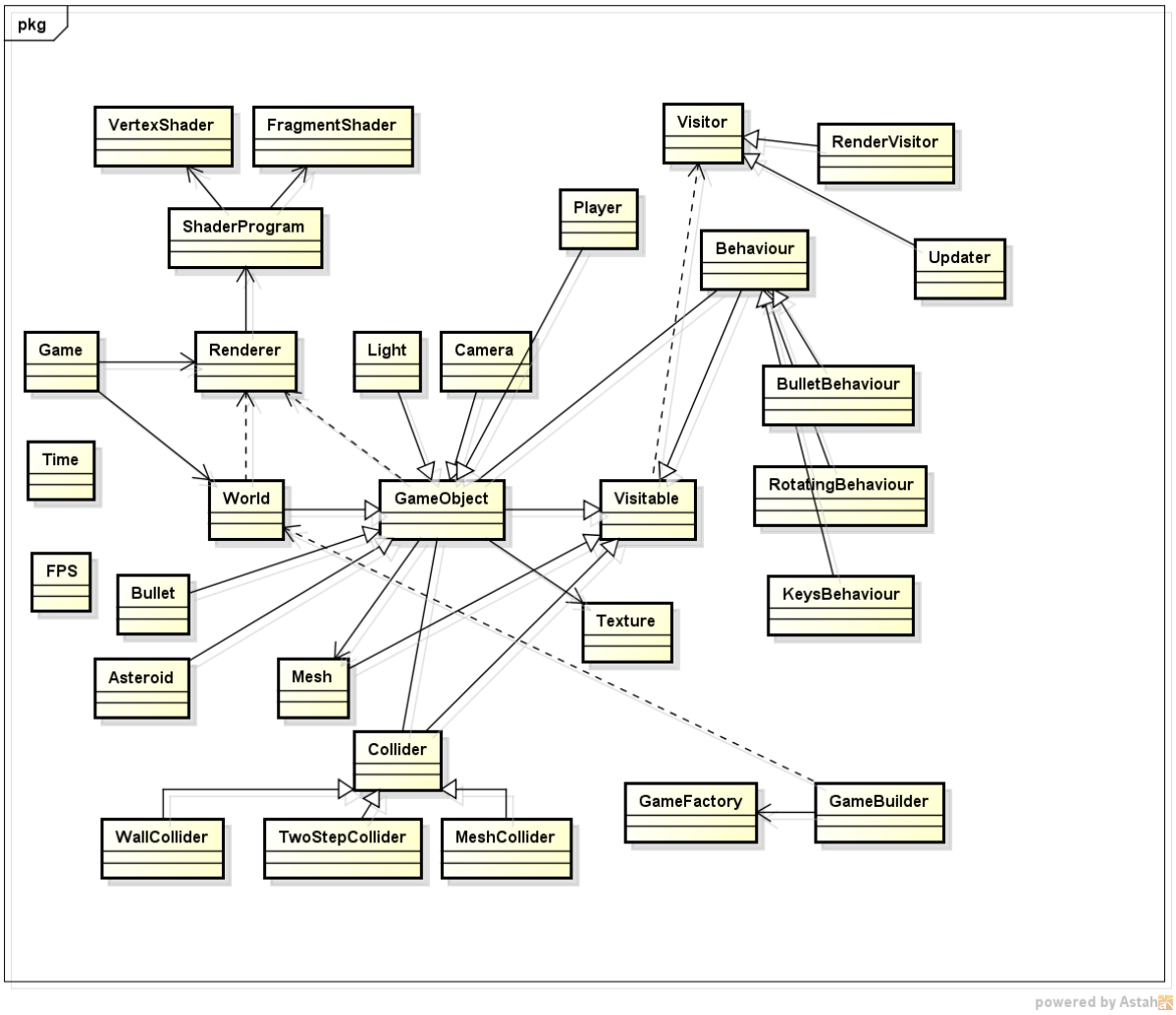
Composite

We gebruiken het composite pattern voor onze gameobjecten. Dit zorgt ervoor dat we eenvoudig gameobjecten kunnen nesten en geen onderscheid hoeven te maken tussen containers en “normale” objecten.

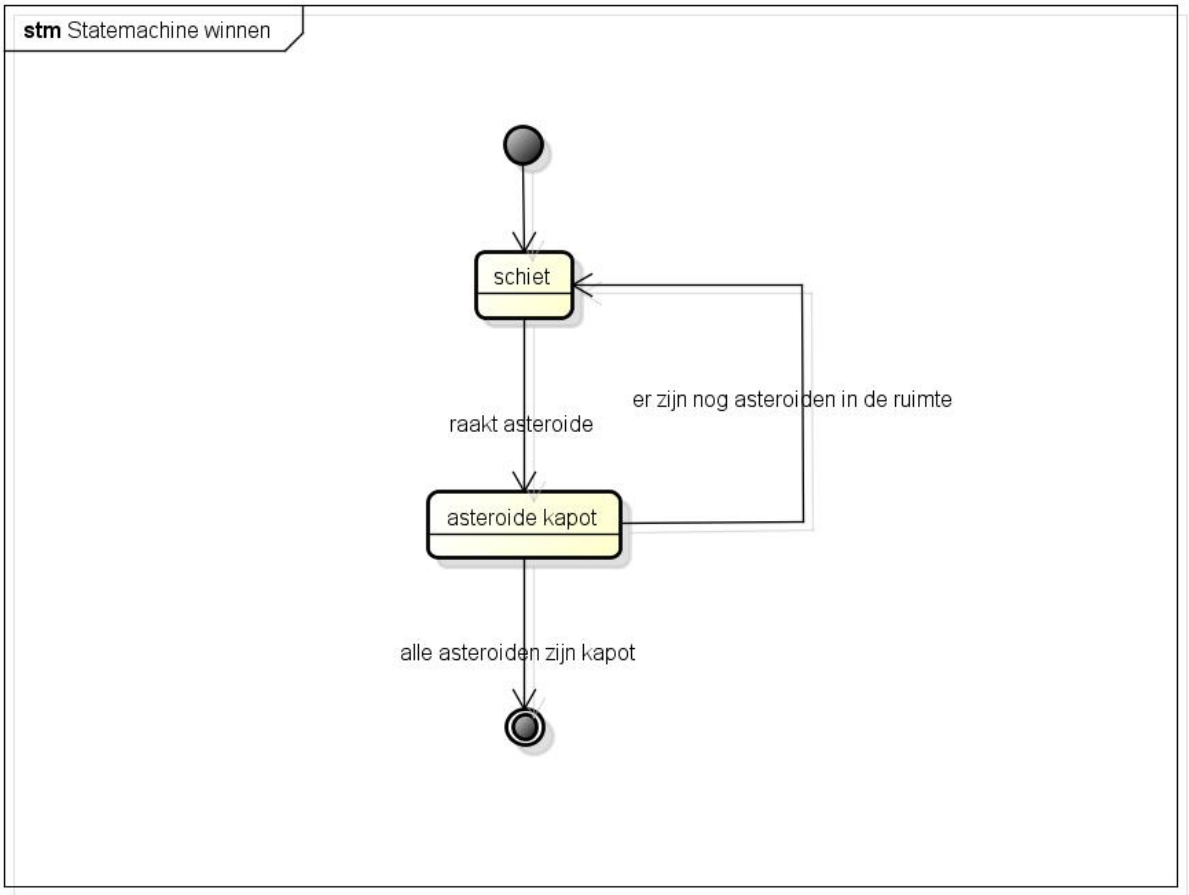
Ontwerp

Ten opzichte van de game voor scene graphs and patterns hebben we de volgende toevoegingen / wijzigingen. We hebben extra colliders toegevoegd: WallCollider, wat een niet bewegende axis aligned bounding box is. MeshCollider wat gebruikt wordt voor NarrowPhase Collision detection met het ruimteschip. En TwoStepCollider, wat een simpele inprecieze collider meekrijgt die (teveel hits geeft, maar niet te weinig) en een duurdere collider. Als de goedkope collider een collision detecteert wordt de duurdere pas aangeroepen. We gebruiken dit met onze Sphere Collider en MeshCollider voor collision Detection met het ruimteschip. Verder hebben we de klasses Asteroid en Bullet toegevoegd voor onze Asteroids en Bullets. Verder hebben we 3 behaviours toegevoegd voor de speler, de asteroids en de bullets.

Class Diagram



State Diagram



Specifieke oplossingen

Camera movement

Om de camera op een juiste manier het ruimteschip te laten volgen hebben we de camera op een vaste afstand op de omtrek van een cirkel met als middelpunt het ruimteschip. De locatie van de camera is dan gebaseerd op de rotatie van het ruimteschip.

De positie van de camera kan berekend worden met de volgende formules:

float camerax = cameradist * sin(yrot * PI / 180) + xpos;

float cameraz = cameradist * cos(yrot * PI / 180) + zpos;

Waar yrot de rotatie om de y-as is van het ruimteschip. En xpos en zpos respectievelijk de x en z positie zijn van het ruimteschip.

Bron en veel uitgebreidere uitleg: http://www.gamedev.net/page/resources/_/technical/game-programming/a-simple-third-person-camera-r1591

Collision Detection en Response

Zodra een asteroide botst met een andere asteroide of onze muur in de ruimte, draaien we de velocity (snelheid + richting) om. Dit leverde eerst problemen op, omdat objecten vast blijven zitten in elkaar, waardoor ze constant om hun velocity draaiden. Om dit te voorkomen hebben we gekozen

om bij te houden met welk object de laatste botsing was om zo maar 1x op deze botsing te reageren. Nadeel hiervan is echter wel dat een botsing tussen meer dan 2 objecten tegelijk nog steeds problemen op kan leveren. Een alternatieve oplossing is om een duwtje te geven in de goede richting, maar omdat je hier voor een minimale snelheid introduceert hebben we voor onze oplossing gekozen.

Broad / Narrow Phase Collision Detection

Voor ons ruimteschip gebruiken we een combinatie van 2 colliders. Een SphereCollider die ons ruimteschip helemaal bevat. En een MeshCollider die heel precies is, en ook weergeeft waar het ruimteschip geraakt wordt. Als de SphereCollider een collision detecteert wordt gekeken of de MeshCollider dit ook doet. Deze wordt dus alleen aangeroepen voor objecten die binnen de sphere zijn. Waardoor deze dure MeshCollider niet zo vaak wordt aangeroepen.

Winnen en verliezen van het spel

Om er uiteindelijk een spel van te maken hebben we ervoor gezorgd dat je dus kunt winnen door alle asteroiden kapot te maken, zonder dat je zelf ook maar één keer wordt geraakt. Als een asteroide het schip raakt, dan is het spel afgelopen en stopt de tijd. De ruimteschip zelf wordt niet “opgeruimd”, zoals dat wel het geval is met de raketten die je schiet en de asteroiden. De game telt alle asteroiden op die aangemaakt worden. Door de asteroiden kapot te maken gaat de teller naar beneden, zodra de teller op nul staat, dan betekent het dat je alle asteroiden kapot hebt gemaakt en heb je dus gewonnen en stopt de tijd. De tijd wordt in dit geval gebruikt voor je score.

Problemen

- Asteroids die aan de rand van de ruimte geplaatst gaan soms door de muur. We hebben niet precies kunnen achterhalen waardoor dit komt. Maar een workaround is om asteroids een eindje vanaf de “muur” te plaatsen.
- Als asteroids te dicht bij elkaar geplaatst worden en allebei dezelfde kant opgaan, blijven ze in elkaar zitten. Ze draaien namelijk allebei direct om van richting en de collision blijft, maar is al behandeld.

Suggesties

- Het spel kan uitgebreid worden door de asteroiden te laten uitspatten in meerder stukjes, zodra het kapot geschoten is. Hier is nog niet naar de verschillende technieken/algoritmes gekeken.
- De 3e dimensie kan beter gebruikt worden, door bijvoorbeeld niet alleen op links, rechts, voor en achter te kunnen, maar ook omhoog en naar beneden (geldt ook voor de asteroiden). Hier hebben we naar gekeken, maar kwamen toen niet uit met de camerabesturing.
- Een puntensysteem in plaats van of toegevoegd met de tijd. Bijvoorbeeld door punten te krijgen van de asteroiden die kapot geschoten zijn (hij kleiner de asteroide, hoe meer punten je krijgt).
- Vanwege tijdsgebrek hebben we geen meerdere levels. Wel kunnen we meerdere levels laten inladen, maar je gaat met de huidige versie niet naar een nieuwe level, zodra je de eerste

hebt verslagen. Op dit moment kun je dus meerdere levels laten inladen door extra level.txt bestanden aan te maken, waar de asteroiden in staan en het programma opnieuw op te starten. Ook kun je alleen nog hardcoded de “ruimte” groter maken. Ideaal gezien moet je dus automatisch naar de volgende level kunnen gaan, waarbij je dan ook in een groter “ruimte” bevindt.

- Een mogelijkheid is om de ruimteschip wat langer te laten leven in plaats van in één keer dood gaan. Bijvoorbeeld pas na drie keer botsen met een asteroide.

Conclusie

Onze engine die we aangepast van het orgineel hebben in SG&P is goed geschikt om de doolhof uit SG&P om te bouwen in een 3rd-person space shooter spel. Hierdoor hebben we zelf heel weinig hoeven te veranderen aan de engine zelf, maar hoefden we eigenlijk alleen maar de “doolhof” om te bouwen naar een “space shooter”. Ondanks dat er veel mogelijkheden zijn met collision detection en er heel veel algoritmes zijn, hebben we besloten om het zo eenvoudig mogelijk te maken om, in de hoop, later dit uit te breiden naar ingewikkelder manieren. Echter viel het tegen en hebben we uiteindelijk toch een redelijk makkelijke algoritme gevonden om onze spel te maken, doordat we ook maar op een “2D”-achtig manier vliegen en schieten, hebben we ook niet veel ingewikkelde algoritmes nodig. Uiteindelijk voldoen we wel aan alle requirements om er een speelbare spel te maken waarin collision detection in zit verwerkt.

Persoonlijke reflectie

Emiel

Ik vond deze opdracht moeilijker dan verwacht. Ik heb daardoor wel veel geleerd. Maar helaas was onze planning niet op deze moeilijkheid en een tegenslag van ziekte gebaseerd. Ik was de laatste week dat we eraan konden werken half ziek, maar gelukkig nu weer helemaal beter. We hebben het meeste werk gedaan gekregen vlak voor en na de deadline. Gelukkig hebben we wat meer tijd gekregen, anders was het lang niet af. De samenwerking ging prima, we konden goed taken verdelen zonder op elkaar te hoeven wachten. De game is al weer een stuk verder dan de scene graphs and patterns game. Het begint al meer op een echte game te lijken.

Voor een volgende keer dus niet zo traag starten, want als je het onderschat dan is je planning moeilijk aan te passen.

Robbert

De opdracht viel tegen qua moeilijkheid in de zin van, dat het moeilijker is dan gedacht. We hoorden verhalen dat het wel allemaal mee viel vergeleken met SG&P, maar dat is dus echt niet zo. Ik vond dit de lastigste. Mijn kennis van C++ is niet zo goed dat we dit zomaar konden maken, ik was eigenlijk al blij dat ik het tentamen C++ eindelijk had gehaald en zelfs nog het mini-project dat in C++ moest. Het maken van een 3D-spel met collision detection is een stuk ingewikkelder dan dat. Ik heb er heel veel over moeten inlezen, maar dan nog vond ik het lastig om het ook daadwerkelijk te implementeren. Uiteindelijk is het toch gelukt er wat van te maken en ik vind wel dat ik meer heb opgestoken van deze drie opdrachten (CG, SG&P en CD) dan van het project wat we een paar jaar geleden hadden gemaakt.