KernModule 1
Game Development
Marlon Sijnesael

Concept:
Naam spel: Scum City
In het spel Scum City is het de bedoeling om een stad te bouwen die zo naar mogelijk is,. Het doel is hier om gebouwen zo te bouwen dat de stad geen infrastructuur heeft en de voorzieningen niet op elkaar aansluiten.
Het originele spel “Sim City” draait om het bouwen van de meest succesvolle stad, mijn twist op dit idee is dat je hier een ongelukkige stad moet maken

Ontwerp:
Om te beginnen heb ik het originele spel proberen te herleiden naar de basis componenten, in dit geval: een grid, gebouwen, UI en een systeem dat de stats van de stad bijhoudt.

om deze componenten te maken heb ik de volgende design patterns gebruikt:

- **KISS** (keep it simple, stupid!) : Ik heb hierbij gezocht naar het simpelste wat kon werken, terwijl ik de code zo generiek mogelijk probeerde te houden. Een voorbeeld hiervan is de duidelijkheid van het grid, waarbij ik de scale van de prefab heb verkleind, zodat het grid zichtbaar wordt voor de speler.
- **DRY** (don’t repeat yourself)
- **SINGLETON:** van sommige objecten mag er maar 1 zijn, zoals de game manager of de score. Door een singleton te gebruiken zijn deze functies en variabelen makkelijk bereikbaar en kunnen alle objecten bij globale informatie, zoals de huidige world state (buildmode of niet). Ik heb wel geprobeerd het gebruik van singletons (en statics te vermijden, zodat niet alle functies en variabelen vanaf ieder punt aanspreekbaar zijn).
- **Abstract factory:** Deze heb ik gebruikt voor de gebouwen in mijn spel. Hoewel de gebouwen andere eigenschappen en gedrag hebben, kan je deze toch voor een groot deel generaliseren. Alle gebouwen hebben een prefab, naam, aantal mensen dat ze huisvesten en een impact op het geluk. Het gedrag van deze gebouwen is echter anders en daarom heb ik een abstract class gemaakt genaamd “gebouw” dat deze generieke eigenschappen / gedrag alvast vaststelt. dit scheelt code repetition en zorgt voor standaard binnen classes voor gebouwen.

Veranderingen in het ontwerp
Er zijn ook een aantal dingen veranderd aan het ontwerp:

Mijn plan was eerst om het observer pattern meer te gebruiken, ik merkte echter dat ik van te voren te veel ging nadenken over welk pattern ik ging gebruiken en daar mijn ontwerp omheen ging bouwen, ipv een ontwerp maken en daar de juiste patterns voor gebruiken. Hierdoor duurde het langer voor mij om een start te maken aan het project en raakte ik ook gedemotiveerd.

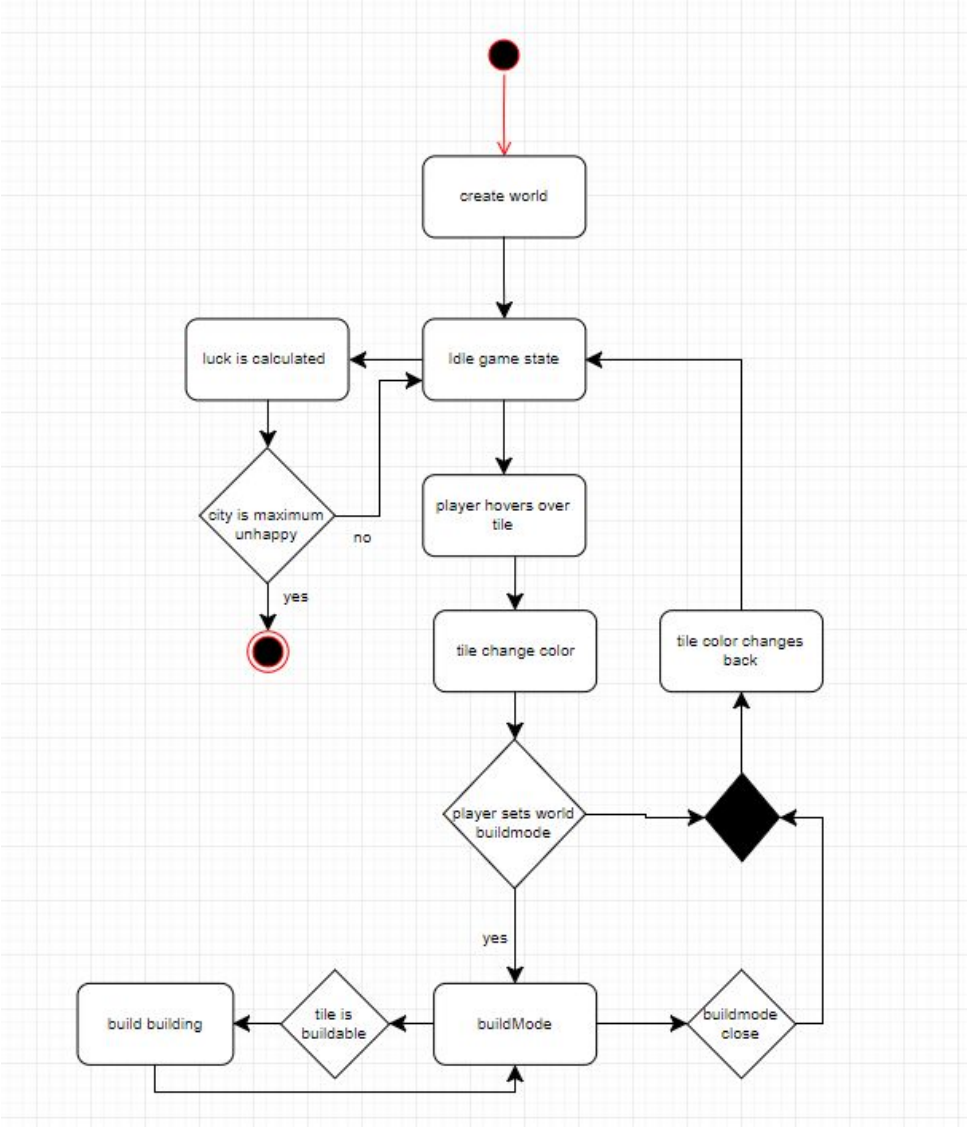
Hiernaast zijn er door tijdgebrek enkele features niet af. Een voorbeeld hiervan is het luck system, de gebouwen doen nu niet veel meer dan er gewoon zijn en hierdoor mist een redelijk groot deel van het spel.

Dingen die anders zouden gaan de volgende keer:

1. Door mijzelf teveel in te perken met design patterns (zoals hierboven genoemd), kwam ik vast te zitten en daalde mijn motivatie voor deze opdracht. De volgende keer ga ik beginnen vanuit wat ik wil maken en eerst nadenken over alle mogelijkheden.
2. Door een week ziek zijn, zijn enkele functionaliteiten niet af. Het geluks-systeem werkt niet zoals ik dat zou willen en de game is nog niet in een staat waar de speler kan winnen of verliezen hierdoor. De volgende keer ga ik proberen beter te plannen zodat ik meer speling overhoud richting de deadline.

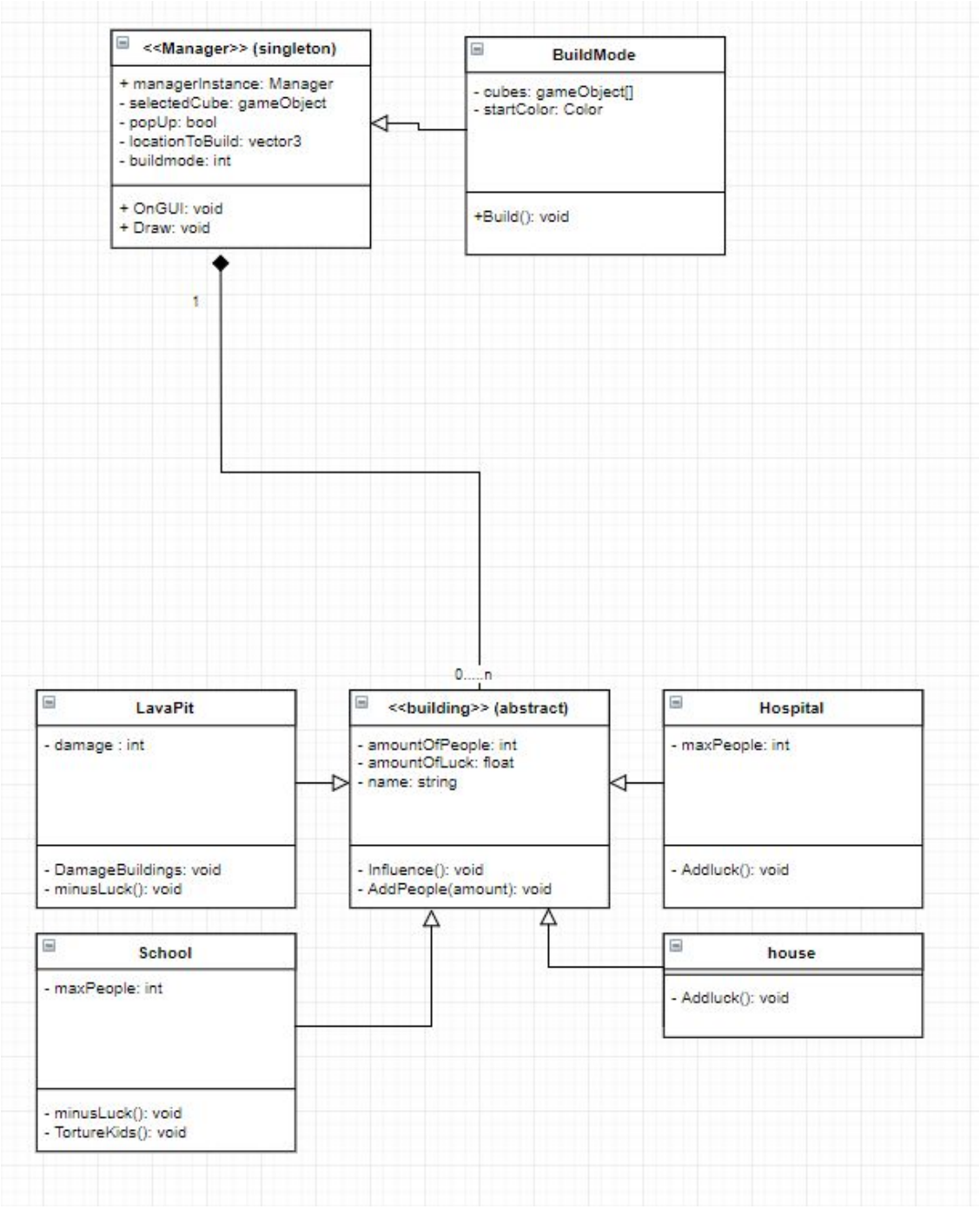
UML:

Activity Diagram:



Toelichting activity diagram:
In het spel zijn 2 states: Idle en Buildmode. Als het spel opstart wordt het grid gegenereerd en wordt de world op idle gezet. Als de speler een van de kubussen met zijn muis selecteert zal de geselecteerde kubus rood worden. Als de speler op de knop “buildMode” drukt dan zal de wereld in buildmode gaan en kan de speler, wanneer hij op een bouwbare kubus klikt, gebouwen plaatsen in de wereld. Als de speler buildmode afsluit zal hij/zij terugkeren naar de idle state.
Als de stad de maximale ongelukkigheid heeft bereikt, heeft de speler het spel gewonnen.

class diagram:



Toelichting class Diagram:

Wegens tijdgebrek is mijn class diagram niet helemaal compleet. In de huidige UML staan wel enkele van de belangrijkste classes. Er is te zien dat “buildMode” inherit van Manager, het gaat hierbij om de “buildMode- int” om te zien of er gebouwd kan worden. Daarnaast is er een overzicht van de abstract factory te zien: alle 4 de buildings erven van <<building>>,

ook is er te zien dat een Manager meerdere buildings kan hebben en kan bestaan zonder buildings, maar buildings kunnen niet bestaan zonder manager.