 Opdracht: Ski-game in MonoGame 

Voor deze opdracht bouw je van nul een kleine ski-game in MonoGame.

Je krijgt een aantal sprites en een sprite font. Alle structuur, logica en architectuur ontwerp en implementeer je zelf. Als je andere sprites wil gebruiken, mag dat, zolang het spelconcept behouden blijft.

Spel concept

De game die je maakt speelt zich af op een verticale (portrait) skipiste die voortdurend naar beneden scrolt. De speler bestuurt één of meerdere skiërs die over de piste glijden en obstakels proberen te ontwijken. De wereld beweegt, objecten verschijnen geleidelijk in beeld en verdwijnen weer, en de speler moet de situatie kunnen inschatten. Het is een eenvoudig concept, maar het spel moet functioneel aanvoelen: skiërs moeten reageren op input, obstakels mogen niet uit het niets opduiken, en botsingen moeten gevolgen hebben.

De opdracht vereist een klein maar volledig spel, bestaande uit meerdere schermen.

Menu

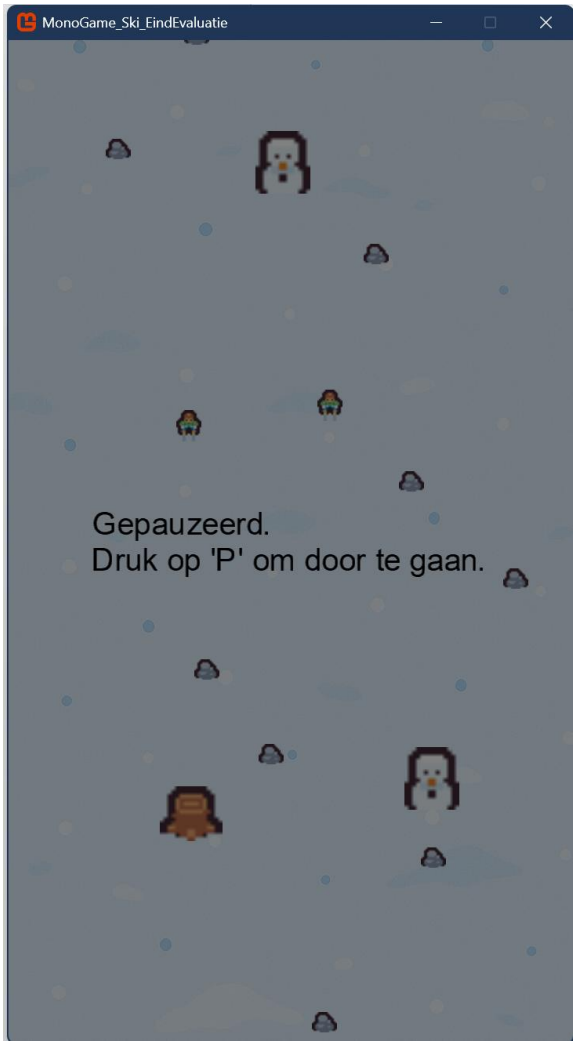
```
Druk 1: 3 Skiiers
Druk 2: Vijanden / 1 skiier
Druk 3: Vijanden / 3 skiier
Druk 4: Vijanden / 2 spelers
```

Bij het opstarten komt de speler in een menu terecht. Vanuit dit menu kan het spel gestart worden. Een eenvoudig scherm dat reageert op toetsten is voldoende.

De speler moet er een keuze kunnen maken uit vier spelmodi:

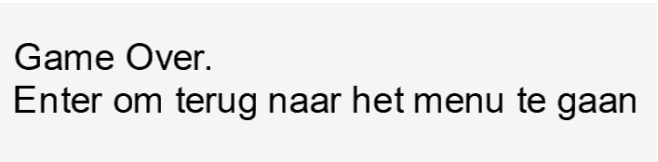
1. Skiën zonder vijanden
   enkel de achtergrond en de skier(s)
2. Skiën met obstakels én één skiër
3. Skiën met obstakels én drie skiërs tegelijk
   je bestuurt ze allemaal met dezelfde input
4. Skiën met met obstakels en 2 verschillende personen
   één gebruikt de toetsen links/rechts/onder/boven, de andere gebruikt qzsd.

Pauze



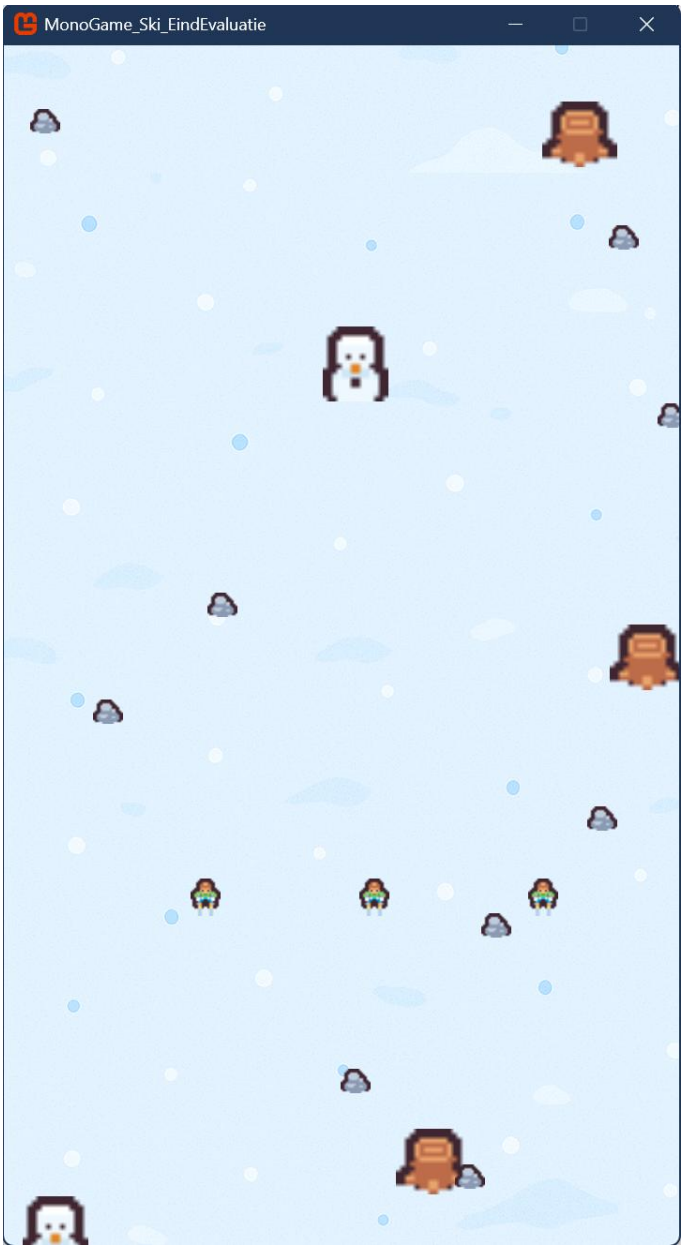
Tijdens het spelen moet de speler het spel ook kunnen pauzeren door op ‘p’ te drukken. Het pauzescherm toont het spel “bevroren”: je ziet dus nog steeds de huidige frame, maar de actie staat stil. Daarna kan de speler terugkeren naar het spel of het beëindigen. Wanneer we op het pauze scherm ‘p’ indrukken dan hervat het spel.

Game Over



Wanneer alle skiërs dood zijn, verschijnt een game-over-scherm met een eenvoudige boodschap en de vraag om op Enter te drukken om terug te keren naar het menu.

Skiën (spelen)



Obstakels

De skipiste bevat drie soorten obstakels: twee statische obstakels (een rots en een stuk boomstam) en een horizontal bewegend obstakel.

- Rots: *de skiër struikelt en valt een stukje achteruit (gewoon de y-richting aanpassen). Spawned elke 500 tot 2.000ms.*
- Boomstam: *Bij aanraking sterft de skier, maar de boomstam blijft bestaan. Spawned elke 1.500ms tot 5.000ms.*
- Sneeuwman: *Deze beweegt van links naar rechts, maar houdt zijn verticale positive op de achtergrond. Eens hij de rand van het scherm bereikt, dan gaat hij*

in de andere richting. Bij aanraking sterft de skier & de sneeuwman. Spawned elke 3.000ms tot 10.000ms.

Denk na over hoe je deze obstakels laat verschijnen. We willen geen GOD-klasse, het is tenslotte een eindevaluatie voor software ontwerp …

Sterven

Bij drie skiërs betekent een dode skiër niet dat het spel stopt, je gaat gewoon verder met de overblijvende skiërs.

Bewegen

De skiër kan naar links, rechts, boven en onder bewegen.
Wanneer de speler geen toets indrukt, beweegt de skiër trager dan de scrollende achtergrond en lijkt dus langzaam naar boven geduwd te worden: alsof hij snelheid verliest en de piste hem voorbijschuift.

SoftwareONTWERP

Hoewel het spel duidelijk moet werken, ligt de echte klemtoon van deze opdracht op softwareontwerp. Je toont dat je:

- het domein zinvol modelleert in een klassendiagram
- verantwoordelijkheden verdeeld hebt over meerdere klassen (dus geen alles-in-Game1-aanpak, geen GOD-klassen)
- patronen en principes inzet om de structuur uitbreidbaar, flexibel en begrijpbaar te houden

Hoe je dat aanpakt, is aan jou. Je zult onder meer moeten nadenken over:

- hoe je de verschillende speltoestanden organiseert (menu, spelen [no enemies, 1 skiier, 3 skiiers & versus], pauze, game-over)
- hoe je spelers, obstakels en de achtergrond structureert en laat samenwerken
- hoe je objecten creëert zonder overal new te gebruiken
- hoe je input behandelt, zodat het coherente en herbruikbare code vormt
- hoe je assets laadt en beheert zodat je Content.Load() niet verspreid door de code staat.

Patronen (Patterns)

Je mag patronen gebruiken die we in de lessen gezien hebben (state, facade, service, manager, factory, …), alsook anderen die je kent.

Hou in het achterhoofd dat het geen checklist is. Forceer er geen patronen in, maar kies degenen die nuttig zijn om flexibiliteit, uitbreidbaarheid en duidelijkheid te ondersteunen.
Aan het einde moet je kunnen uitleggen waarom jouw ontwerp eruitziet zoals het eruitziet, waarom bepaalde klassen bestaan, en waarom die op die manier gekoppeld zijn.

Inleveren

Je

- vult het formulier in (https://forms.cloud.microsoft/e/7mbRFJNGB7)
- zet je app + diagram op **GitHub**
- geeft me (**tommy.uytterhaegen@hogent.be**) **toegang**
- stuurt de GitHub link, nogmaals, als inzending op de opdracht

Hier wil ik een werkend spel terugvinden, op een leesbare & logische manier, alsook een diagram van je ontwerp. Zorg ervoor dat jouw code & ontwerp met elkaar kloppen …

Heel **belangrijk** is ook dat je beknopte commentaar zet bij relevante code waarin je je keuzes toelicht. Als je er niet in slaagt je keuze uit te leggen, wil dit meestal zeggen dat je deze niet bewust gemaakt hebt.

Opgelet: Tijdens de mondelinge verdediging moet je kunnen aantonen dat je jouw oplossing zelf hebt opgebouwd en de onderliggende concepten begrijpt, alsook eventuele niet gebruikte concepten uit de les.

Als blijkt dat je je oplossing niet zelf hebt opgebouwd, dan zal dit **automatisch** leiden tot een **onvoldoende**.

Het doel van de verdediging is net om te evalueren of je de geziene stof beheerst, overtuig me dus dat jij het ontwerp gemaakt hebt en de implementatie begrijpt.

O.M.G. Dit is zo tof!

Extra features zijn welkom, maar mogen nooit ten koste gaan van een helder ontwerp. Start vanuit de vraag: “Hoe ontwerp ik een structuur die uitbreidbaar blijft?” en niet vanuit “Hoe krijg ik dit zo snel mogelijk aan de praat?”

TIP: Bekijk (en begrijp) de Surfing Pikachu oplossing die we hadden op het einde van les 3 (https://github.com/tommy-uytterhaegen/HoGent)