Opdracht cp2

Opgave omschrijving

We gaan proberen om een spel te maken (een dungeon crawler). Dit spel heeft een dungeon bestaande uit een reeks aan elkaar verbonden kamers. Deze kamers kunnen monsters, items of niets bevatten. Er is ook 1 kamer met een schat. De speler start aan de ingang en kan steeds kiezen door welke deur hij wil gaan naar een volgende kamer. Als er een item in de kamer is, krijgt de speler een positief effect en als er een monster is vechten ze automatisch. De speler wint als hij de schat vindt. Het volledige spel moet kunnen worden opgeslagen en ingeladen.

Specificaties voor elk onderdeel

Dungeon generation

Een variabele bepaalt hoeveel kamers er moeten zijn. Je moet gebruik maken van een graph structuur met een adjacency list om de kamers aan elkaar te verbinden, startende van de ingang. Meer info over die structuur hier (Graph Data Structure). Elke kamer moet aan 1-4 andere kamers verbonden zijn (dus die waar je uit komt + 0-3 anderen). Je moet je geen zorgen maken over of het fysiek mogelijk is voor alle kamers om naast elkaar te liggen zoals ze verbonden zijn.

De volledige structuur van kamers en verbindingen moet op de heap worden opgeslagen.

Als mogelijke extra : maak gebruik van eigen 4-dimensionale doubly linked list naar de 4 mogelijke buren i.p.v. Graph Data Structure, waarbij NULL gebruikt wordt als er geen toegang is (bedenk een goede oplossing voor opruimen van het geheugen).

De player

De speler heeft een locatie (het id van de huidige kamer), health points (hp) en damage. Wanneer hij tegen een monster vecht verliest het monster hp op basis van zijn damage en verliest hij hp op basis van het monster zijn damage.

Kamer inhoud

Elke kamer heeft 1-4 deuren naar andere kamers. Elke kamer heeft een id en dat id staat op de deur naar die kamer. Verder kan er ook een monster, items of de schat in de kamer staan. Items verbeteren de speler: hp dat je verloren hebt terugkrijgen, verhoogde damage of eender wat je zelf kan bedenken. Een monster start automatisch een

gevecht. De schat eindigt het spel. Voorzie minstens 2 types van monster en 2 types van item. Denk eraan dat je moet bijhouden indien een kamer al bezocht en leeggemaakt is.

Als mogelijke extra : maak gebruik van function pointers om de relevante functionaliteit aan een kamer te verbinden, je kan de function pointer aanpassen naar "reeds bezocht" eens je er al bent geweest.

Bitwise gevecht

Wanneer een monster en een speler starten met vechten wordt er een random getal tussen 0-16 aangemaakt en omgezet in binaire vorm. Bij elke 0 valt het monster de speler aan en bij een 1 omgekeerd. Als aan het einde van een ronde beiden nog leven wordt dit herhaald tot er een winnaar is.

Visualisatie

Het is niet de bedoeling dat de volledige dungeon geprint wordt. Dit zou ook niet praktisch gaan aangezien verbonden kamers niet naast elkaar hoeven te liggen. Wat wel de bedoeling is, is dat er tekst geprint wordt om aan te geven wat de speler allemaal ziet en wat er allemaal gebeurt. Bijvoorbeeld:

“de held gaat naar kamer 5”

“De kamer is leeg en heeft deuren naar kamer 4, 23 en terug naar 12”

“Kies een deur: “23

“De held gaat naar kamer 23”

“Er is een goblin in de kamer en er zijn deuren naar kamer 7 en terug naar 5”

“aanval volgorde: 0110”

“de held verliest 3 hp (12/20)”

“de goblin verliest 5 hp (3/8)”

“de goblin verliest 5 hp en sterft (0/8)”

“kies een deur:”7

…

Opslaan

Ten slotte moet het mogelijk zijn het spel permanent op te slaan en in te laden. Sla het spel op naar 1 of meerdere files en geef de gebruiker de keuze om het recentste spel te hervatten of een nieuw spel te starten wanneer je programma opstart.

Als mogelijke extra : maak gebruik van een correcte json structuur bij het bewaren en inlezen van de gegevens.

Evaluatiecriteria

Voorwaarden

We delen enkel punten uit indien het geheel compileert en je de onderstaande puntjes kan aantonen in je code en juist kan uitleggen. AI-assistentie is toegestaan, zolang jij de code volledig kan uitleggen.

1. Voorzie en gebruik juiste structs voor speler en kamer
2. Gebruik van pointers waar nuttig
3. Gebruik minstens één call by reference nuttig in de code
4. Maak gebruik van argument to main voor het genereren van een aantal kamers (new game) of het laden van een bestaand spel via bestandnaam (load game)
5. Maak gebruik van memory allocatie om de gegevens op de heap te bewaren
6. Voorzie correcte opruiming van memory zodat er geen zombie memory of memory leak ontstaat
7. Maak gebruik van linked lists op de juiste plaats
8. Pas de bitwise operaties correct toe
9. Maak gebruik van file handling om je spelstatus te bewaren in een bestand
10. Maak gebruik van file handling om je oude spelstatus te kunnen inladen van een bestand
11. Voorzie alle code voor het genereren van de dungeon met minstens 2 types van monster en 2 types van item en het uitvoeren van de bijhorende code tijdens het eerste contact alleen (i.e. weet waar je al geweest bent)
12. Zorg dat het spel volledig speelbaar is en niet vastloopt of onwillig afsluit (crash)
13. Schrijf code volgens industriestandaard in handelbare functies en geen globale variabelen gebruiken (clean code)
14. Code staat proper zonder overbodige bestanden op GitHub in meerdere nuttige correcte commits. Dat wil zeggen, verschillende features staan in aparte commits met logische benaming.
15. Voorzie 1 van de 3 voorgestelde extra's (i.e. 4-dimensionale doubly linked list, function pointers, json structuur)

Puntenverdeling op 25

Onderdeel	Omschrijving	Punten
Dungeon generation	Is de structuur opgebouwd zoals gevraagd? Wordt alles correct gealloceerd? Zijn er memory leaks?	8
Kamer inhoud	Zijn elk van de mogelijke effecten binnen een kamer correct geïmplementeerd? Is de functionaliteit volgens een goede, overzichtelijke structuur aangebracht?	6
Bitwise combat	Is het bitwise combat systeem correct geïmplementeerd? Wordt er correct gebruik gemaakt van bitwise operators?	3
Opslaan	Kan het spel zoals gevraagd worden opgeslagen en ingeladen? Is de methode voor het op te slaan voldoende doordacht?	4
Structuur en netheid	Is de code in zijn geheel overzichtelijk en leesbaar? Is er bijvoorbeeld correct gebruik gemaakt van header files, functies, structs,...	3
github	Heeft de student goed gebruik gemaakt van commits om voortgang in zijn of haar project bij te houden. Bij goed gebruik van commits is het verloop van het project te volgen uit de commit history.	1
Totaal:		25