FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

VAKGROEP INFORMATIETECHNOLOGIE
ONDERZOEKSGROEP IDLAB

PROJECT PROGRAMMEREN
SLIME VOLLEY IN C++

Jeroen van der Hooft

Academiejaar 2016-2017

IDLab
INTERNET & DATA LAB

imec

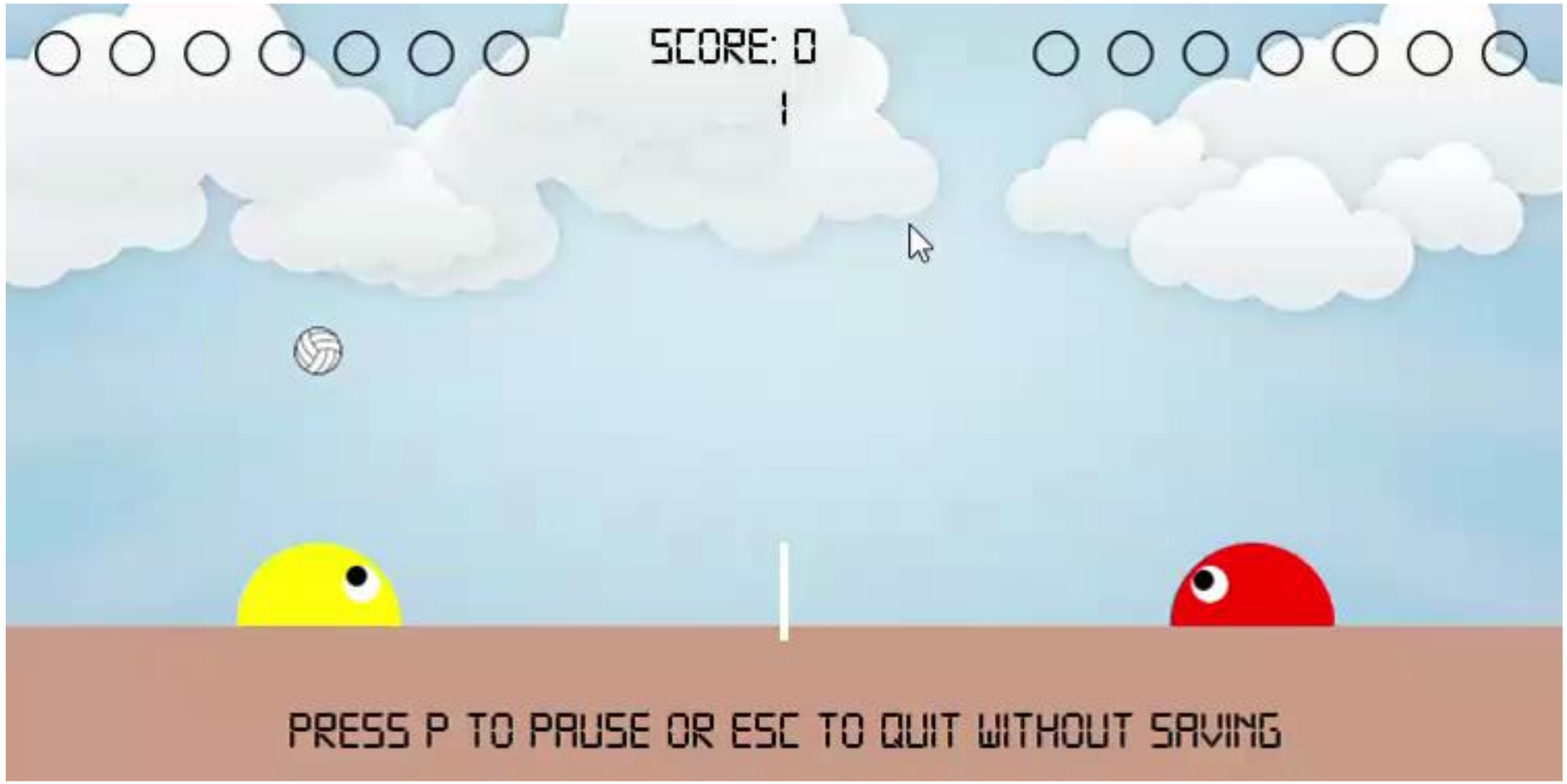
OVERZICHT

1. Het spel: Slime Volley
2. Game programming
3. Entity-component-system framework
4. De Allegro library
5. Praktische afspraken

1. HET SPEL: SLIME VOLLEY

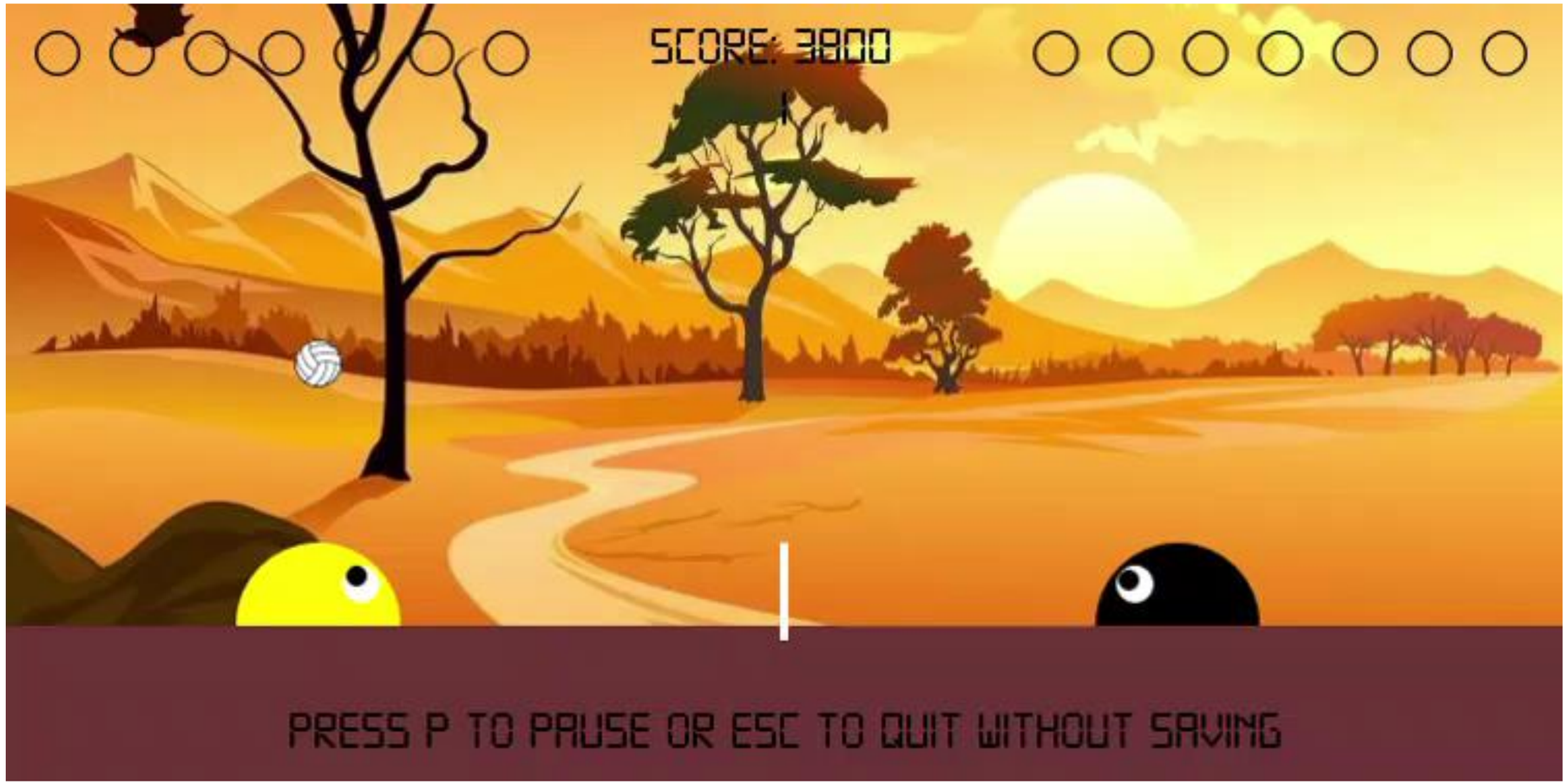
DEMO – MULTIPLAYER

DEMO – SINGLEPLAYER (LEVEL 1)



DEMO – SINGLEPLAYER (LEVEL 2)

DEMO – SINGLEPLAYER (LEVEL 3)



DEMO – REPLAY

2. GAME PROGRAMMING

DE GAME LOOP

De applicatie itereert in een “game loop”

```
while user does not exit do
    handle user input
    run AI
    move objects
    resolve collisions
    draw graphics
end while
```

Bevat de onderliggende logica van het spel

BEWEGENDE BEELDEN

Bewegende beelden worden vloeiender door:

1. Het vergroten van het aantal frames per seconde (FPS)
2. Het verkleinen van het verschil tussen opeenvolgende frames

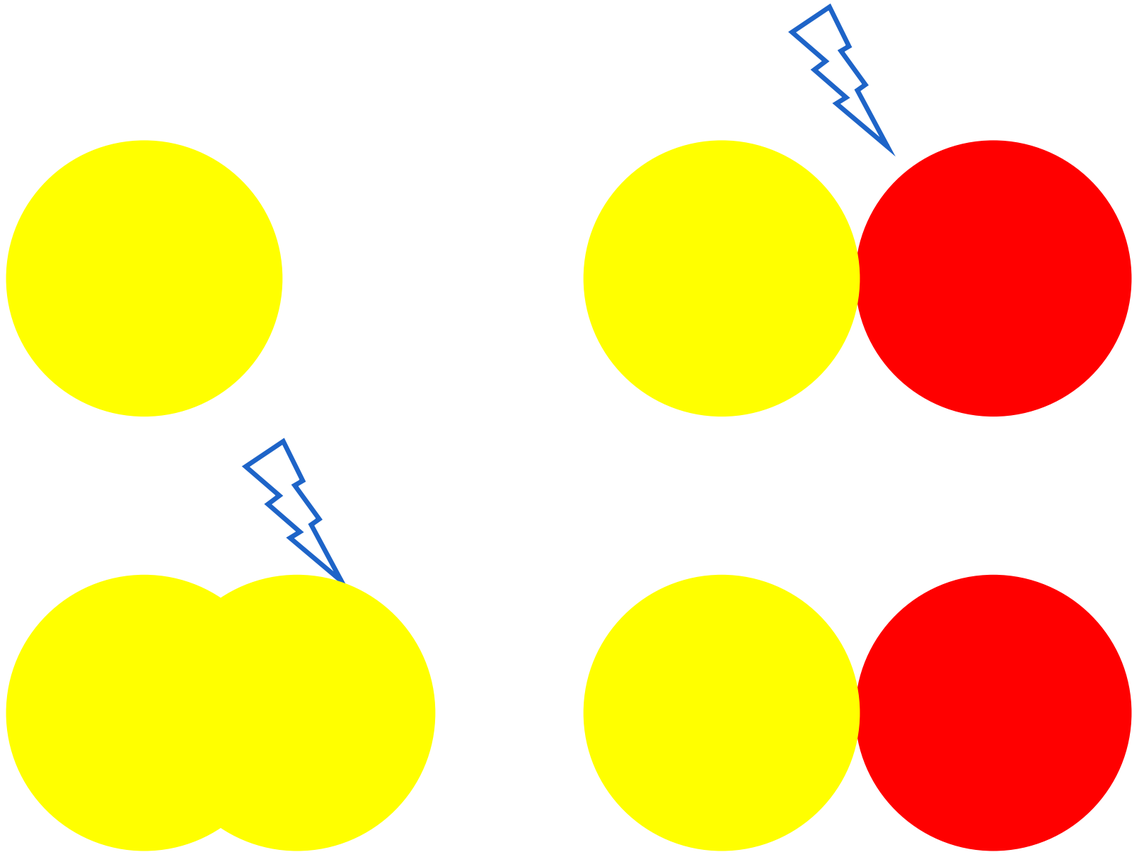
In dit project wordt een frame rate van 40 FPS gebruikt

COLLISION DETECTION

Wanneer objecten verplaatst worden, kunnen botsingen ontstaan

Collision detection behandelt deze:

1. Reactief
2. Pro-actief



3. ENTITY-COMPONENT-SYSTEM FRAMEWORK

ALGEMEEN PRINCIPE

Entiteit:	een “object” in het spel
Component:	een “eigenschap” van een entiteit
Systeem:	interageert met alle entiteiten die over welbepaalde componenten beschikken

EEN VOORBEELD BIJ SLIME VOLLEY

Entiteit: speler

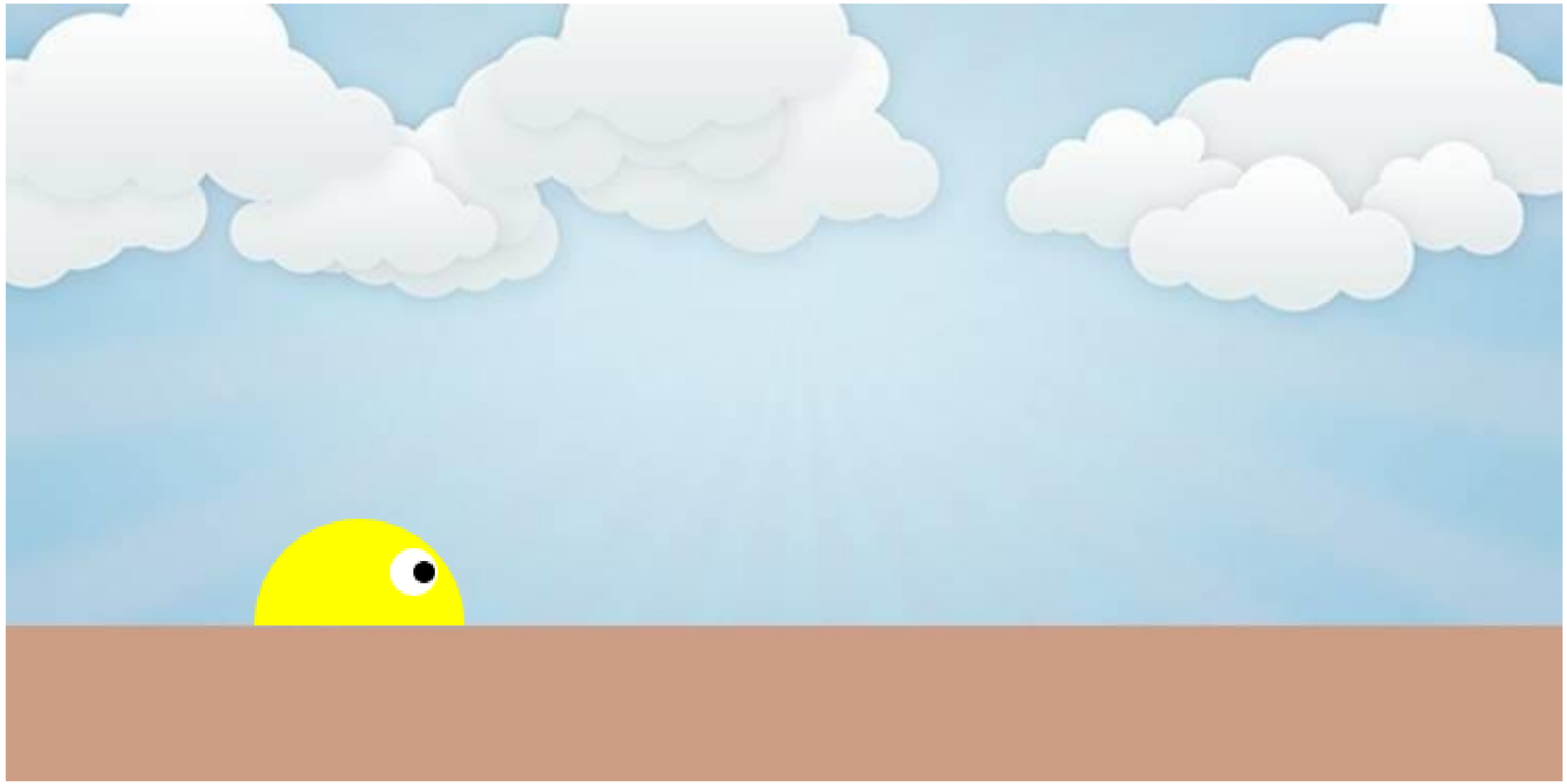
Component: grafisch

afbeelding

positie (x, y)

Systeem: behandelt alle entiteiten die weergegeven moeten worden

EEN VOORBEELD BIJ SLIME VOLLEY



EEN VOORBEELD BIJ SLIME VOLLEY

Entiteit: speler

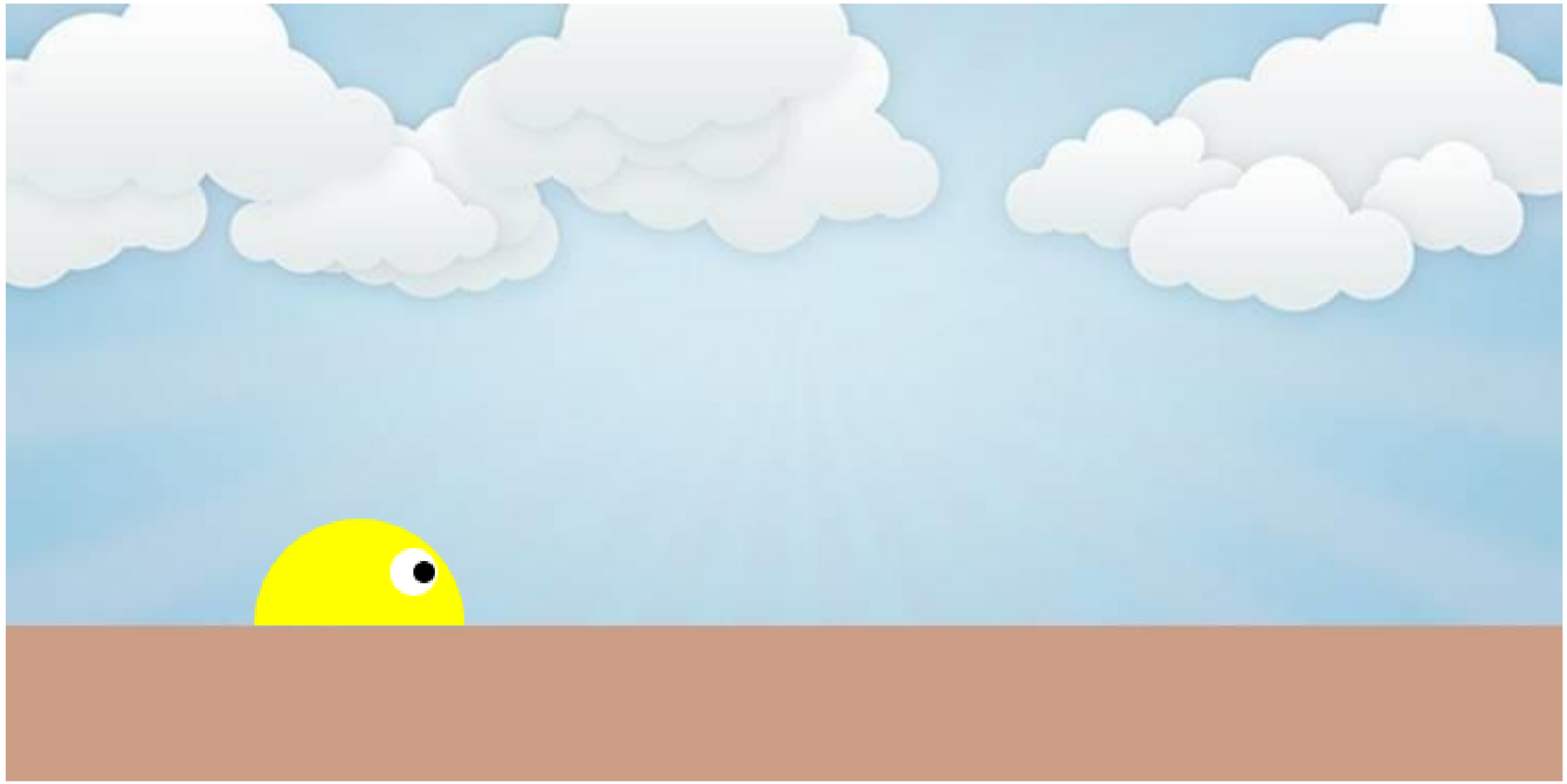
Component: beweging

snelheid (v_x, v_y)

versnelling (a_x, a_y)

Systeem: behandelt alle entiteiten die bewogen moeten worden

EEN VOORBEELD BIJ SLIME VOLLEY



HET FRAMEWORK IS REEDS GEÏMPLEMENTEERD

Details terug te vinden in de opgave (UML-diagram)

Heel wat zaken zijn zelf nog te implementeren:

Entiteiten:	spelers, bal, net…
Componenten:	grafisch, beweging…
Systemen:	gebruikersinput, beweging, collision detection, game state, weergave objecten, AI (!)…
Spelvormen:	multiplayer, singleplayer, replay

ARTIFICIAL INTELLIGENCE (AI)

De AI bestaat uit vier componenten:

1. Opslaan
2. Positioneren
3. Springen
4. Returnen

Twee levels te implementeren (een derde optioneel)

Details uitgewerkt in pseudocode

4. DE ALLEGRO LIBRARY

EEN KORTE TOELICHTING

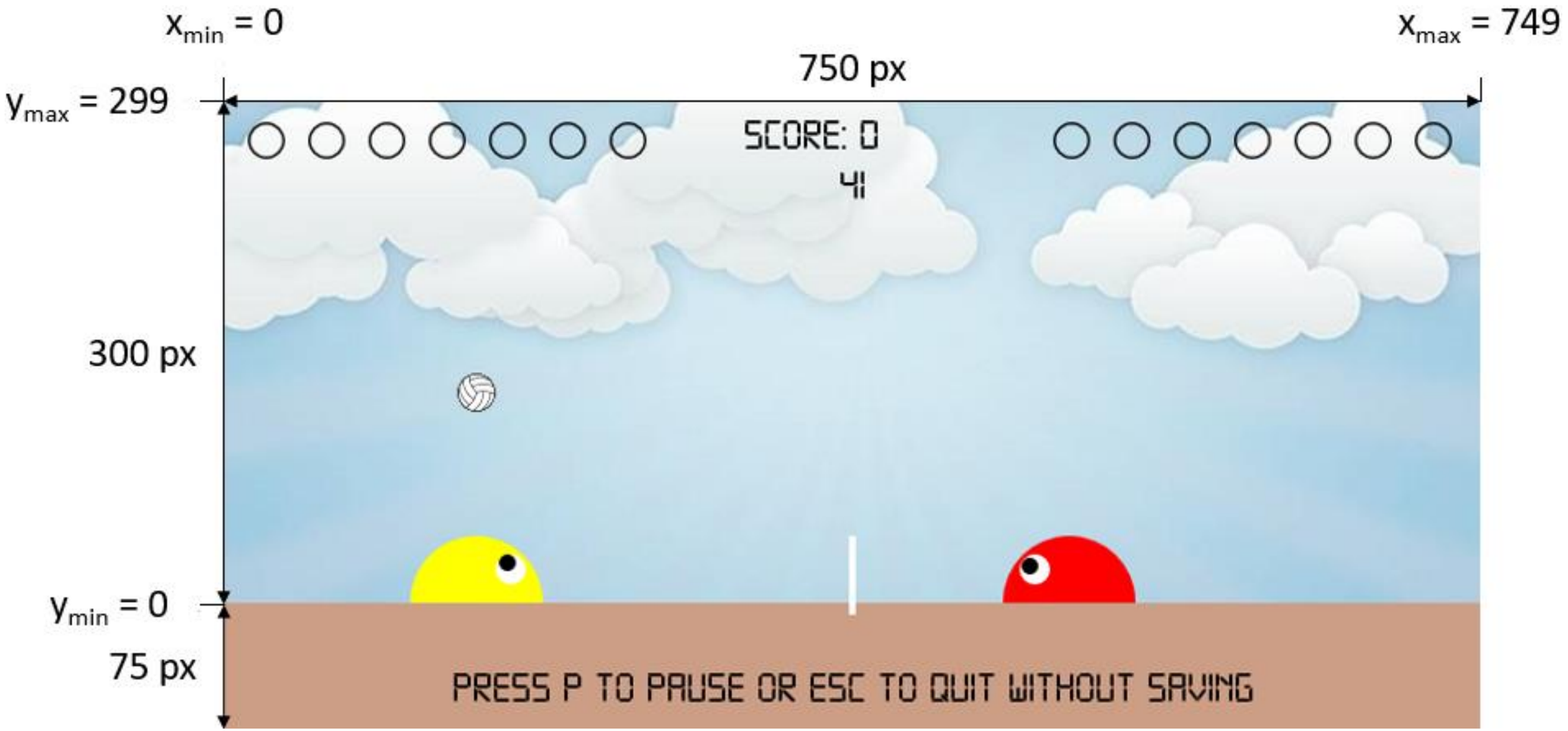
Allegro is een cross-platform library voor video gaming

Handelt taken af op het laagste niveau:

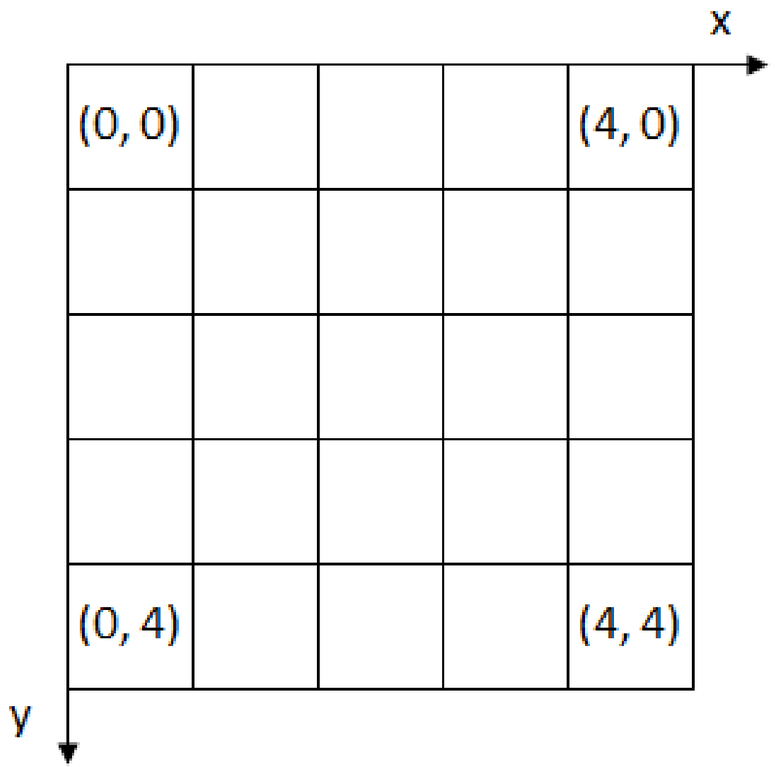
- Registreren van gebruikersinput
- Tekenen van de nodige afbeeldingen
- Afspelen van bepaalde geluiden
- …

In dit project wordt een wrapper gebruikt om deze library aan te spreken

HET ASSENSTELSEL IN HET SPEL (CONVENTIE)

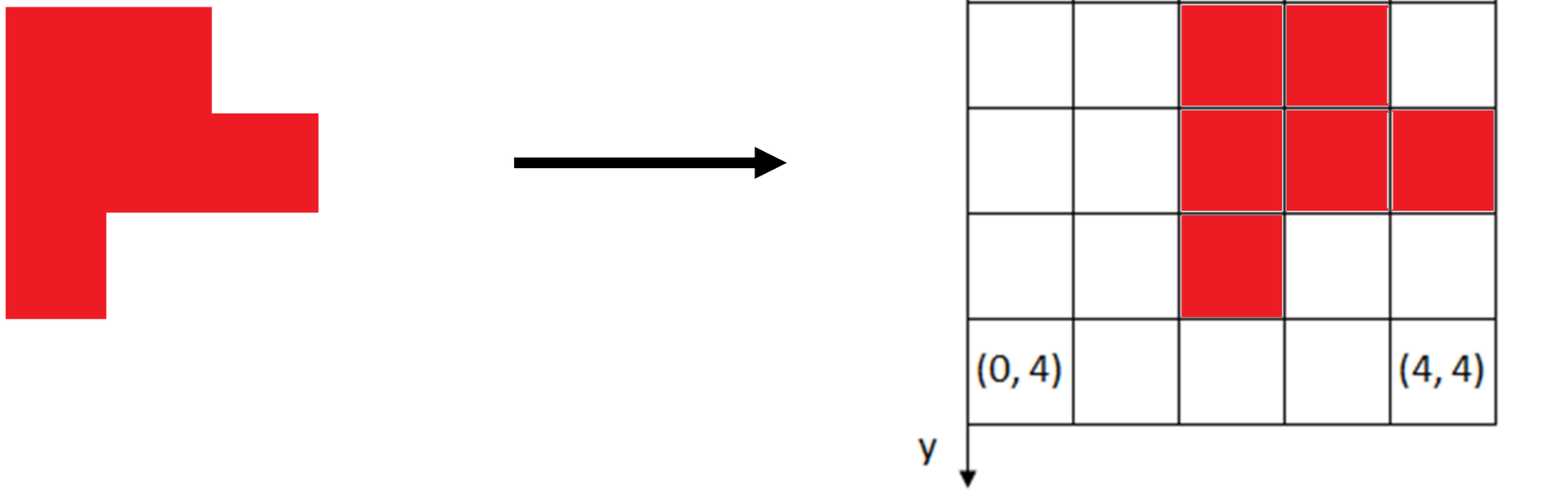


HET ASSENSTELSEL IN ALLEGRO



HET ASSENSTELSEL IN ALLEGRO

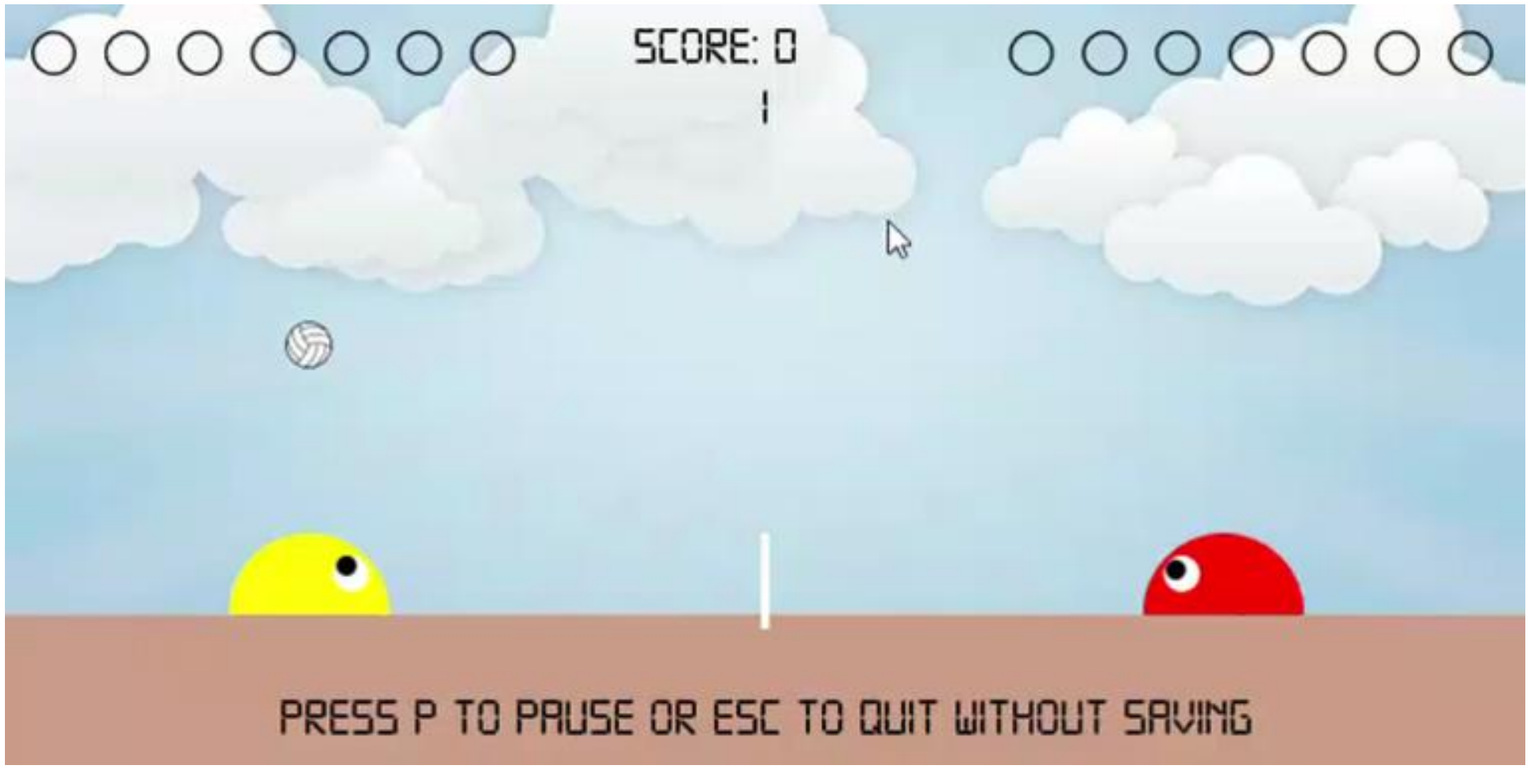
Plaats afbeelding op positie (2, 1)



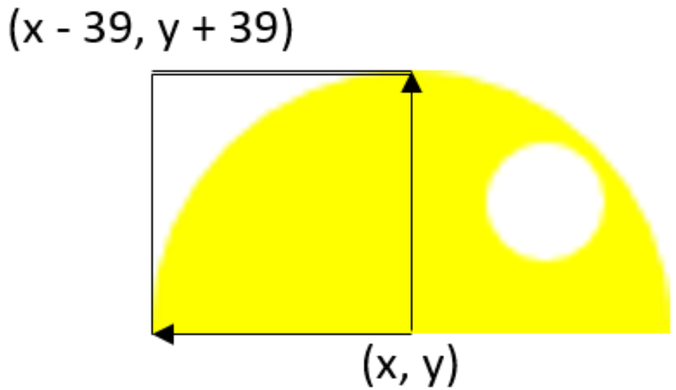
NADEEL: NIET INTUÏTIEF BIJ RONDE VORMEN

De x-waarde voor de speler en de bal zijn verschillend

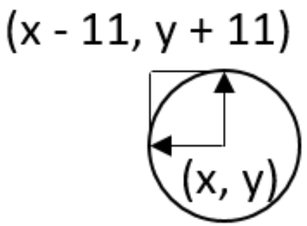
De y-waarde voor een speler in rust is verschillend van 0



OPLOSSING: OFFSETS GEBRUIKEN



x_off = 39, y_off = -39

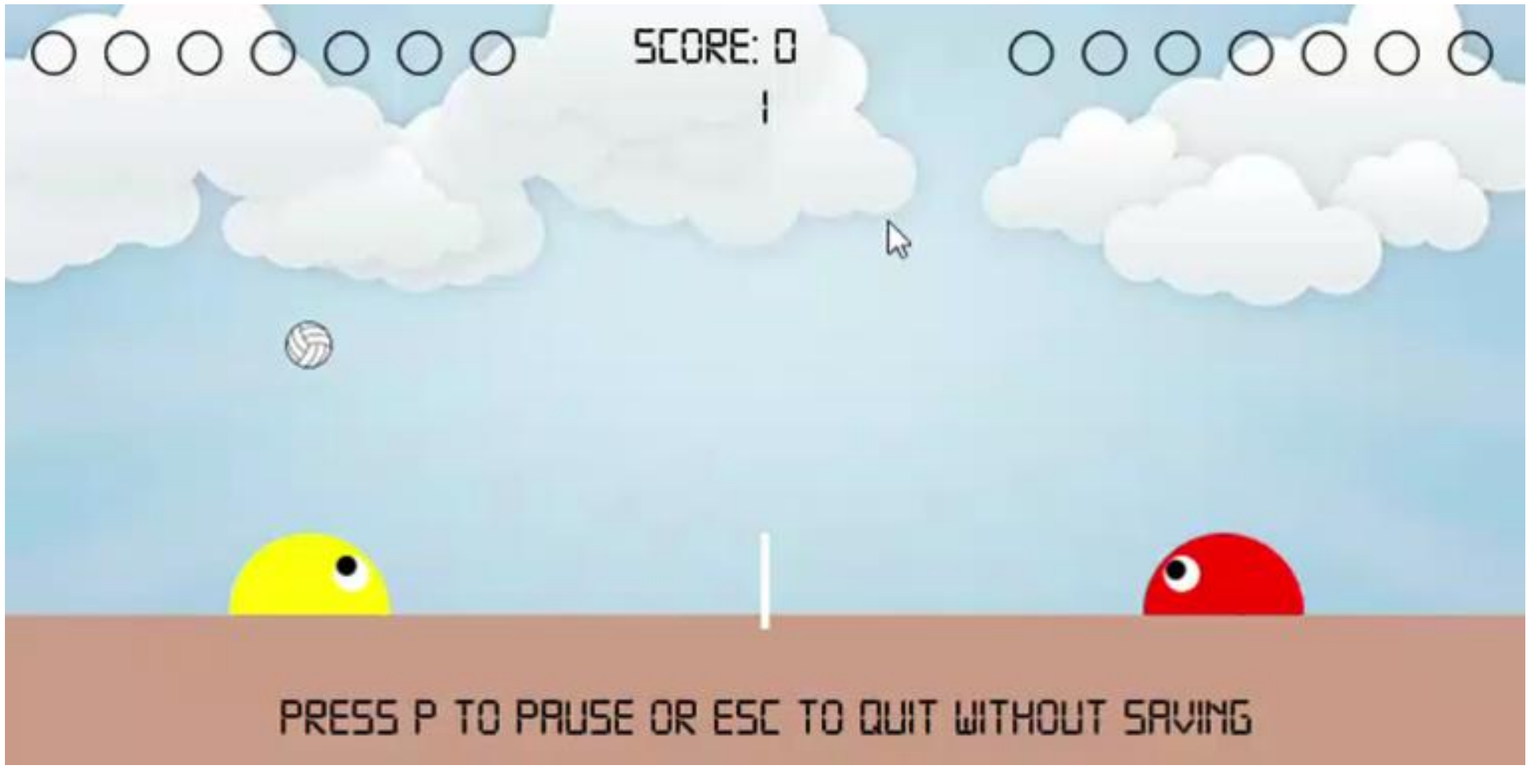


x_off = 11, y_off = -11

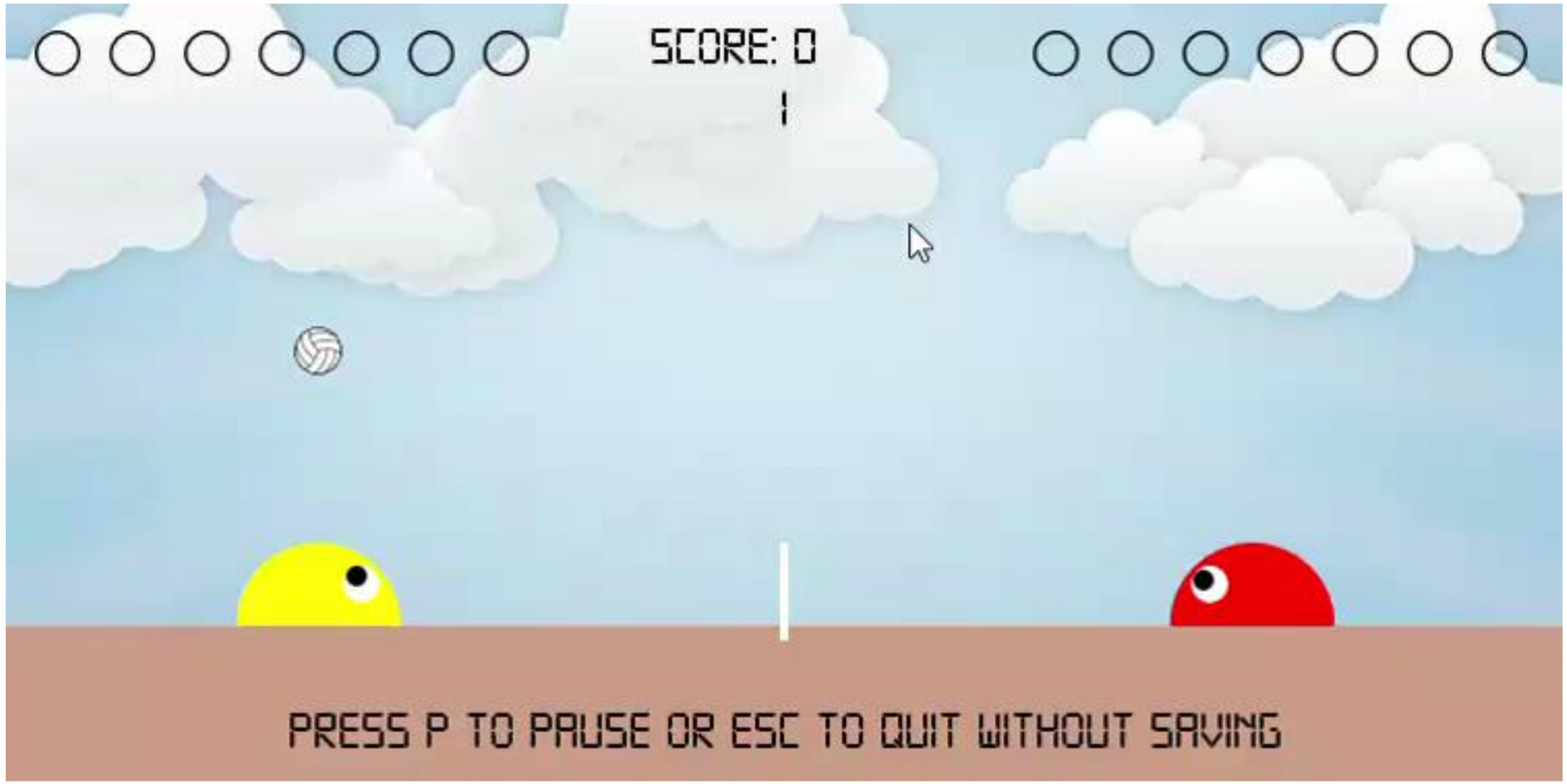
NA TOEPASSEN VAN DE OFFSETS:

De x-waarde voor de speler en de bal zijn nu gelijk

De y-waarde voor een speler in rust is nu 0



DEMO RECAP – SINGLEPLAYER (LEVEL 1)



5. PRAKTISCHE AFSPRAKEN

PRAKTISCHE AFSPRAKEN

Project telt mee voor 3 van de 20 punten

In groepen van twee, in te schrijven op Minerva

Evaluatie op basis van broncode en kort verslag:

- Basisversie van het spel
- Uitbreiding (level 3)

Algemene vragen via pgm@lists.ugent.be

BESTANDEN OP MINERVA

Opgave

Presentatie

Demovideo's

Spelgerelateerde bestanden

- MVS Solution (!)
- Headers en implementatiebestanden
- Assets (afbeeldingen, fonts en highscores)
- Allegro library en properties

BELANGRIJKE OPMERKINGEN

Referentiecompiler: Microsoft Visual Studio 2015

Andere OS's en IDE's zijn toegelaten, maar worden niet door de begeleiders ondersteund

De ingezonden code moet compileerbaar zijn met de referentiecompiler

Een stappenplan werd voorzien in de opgave

Gebruik bij voorkeur GitHub (extra uitleg in tutorial)

Vermijd memory leaks (extra uitleg in appendix)

INDIENEN

Broncodebestanden (.h en .cpp)

- Basisversie en eventuele uitbreiding apart

Verslag (max. 2 pagina's)

- Belangrijkste ontwerpsbeslissingen
- Taakverdeling (wie deed wat?)
- Problemen bij uitvoering (indien van toepassing)

Minerva dropbox naar Bruno Volckaert en Femke De Backere

Deadline: zondag 21 mei 2017, 23u59