Techniques for Designing GPGPU Games

Mark Joselli

Esteban Clua

Agenda

- Presentation;
- Background;
- Motivation;
- Objectives;
- Games and GPGPU;
- Techniques analyzed;
- Examples;
- Conclusions;

Presentation: Mark Joselli

- Chief-Developer-Officer of Nullpointer;
- Senior Research of MediaLab/UFF;
- Professor of games in Fac. CCAA;
- Game Developer;
- Researcher in GPGPU;
- Created the first Framework for GPGPU games;

Presentation: Prof. Esteban Clua, Dr.

- Associate Professor of Universidade Federal; Fluminense – Rio de Janeiro – Brazil;
- Director of MediaLab/UFF;
- CUDA Research Center at UFF;

About: Nullpointer

- Created in 2004, NullPointer acts as a development and innovation lab facility in Rio de Janeiro, working with academic / private research centers;
- We are strongly focused on market requirements, applying the professional experience of our team to real life situations within the retail, telecommunications and financial industries;
- The company has also expanded into a technology consultancy role, especially with regards to GPU computing platforms.

About: MediaLab/UFF

- First CUDA Research Center in Brazil;
- During many years, since the beginning of the concept of GPGPU, we have been researching games and the use of GPGPU with them;
- We are a laboratory of games, scientific simulation and HPC with the use of GPGPU;
- We compose a team of researches who are pHds, masters and graduates;

Motivation

- High processing power of the GPUs, the GPUs are very powerful and games, as real time applications, need its power to add performance;
- Nowadays gaming hardware is able to process tasks in parallel (PlayStation, Xbox, multi-cores CPUs, GPUs…);

Objectives

- Present some techniques to get advantage of the GPUs in games;
- Present a framework for validating the techniques;
- Present a game for validating the framework;

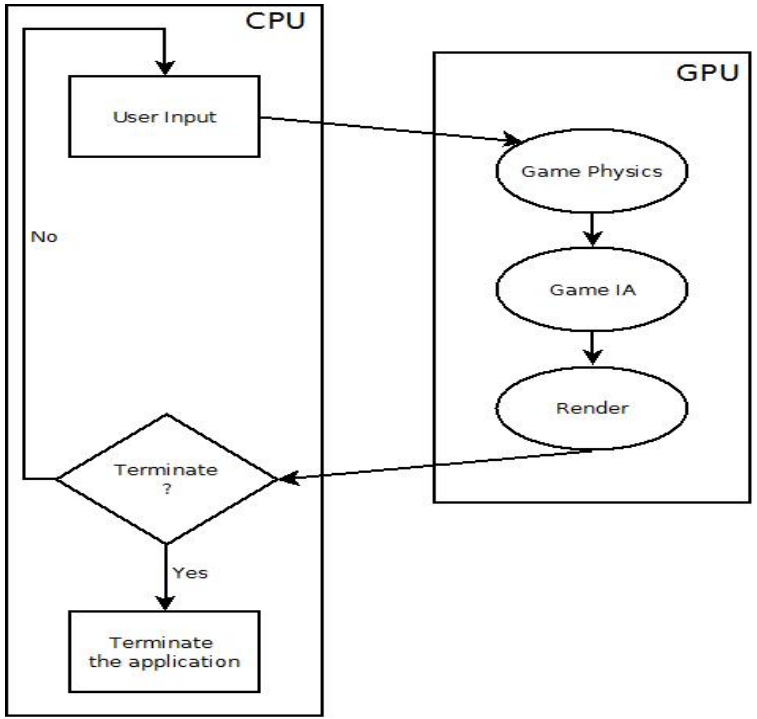
Games and GPGPU

- Most of todays use of GPU in games is resumed in physics (PhysX);
- But also the behavior could be implemented in the GPU;
- As far as we know, there are no projects that deals all the game logic on the GPU;

Techniques Analysed

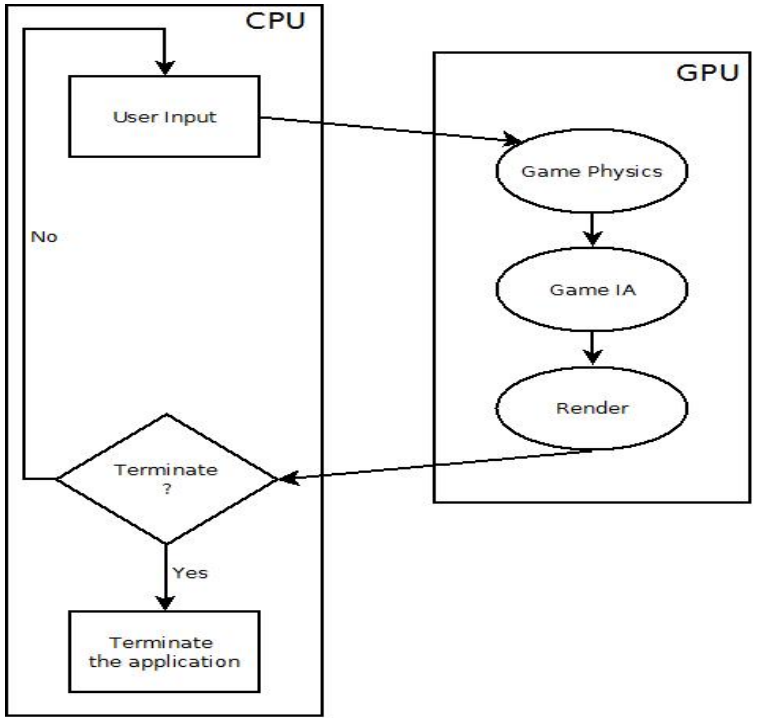
- Avoid CPU-GPU transfer
  - One of bottlenecks of GPU's application is the transfer time between the CPU-GPU;
- Shared Memory
  - The use of shared memory can optimize the GPGPUs kernels in up to 50%;
- Integrate the AI Behavior with Physics
  - Some algorithms that both tasks process are repeated, and should be integrated;

Architecture



- The CPU is responsible for:
  - Windows Creation;
  - Gather input and send it to the GPU;
  - Make the GPU calls;
  - Play sound effects;
  - Finish the application;

Architecture

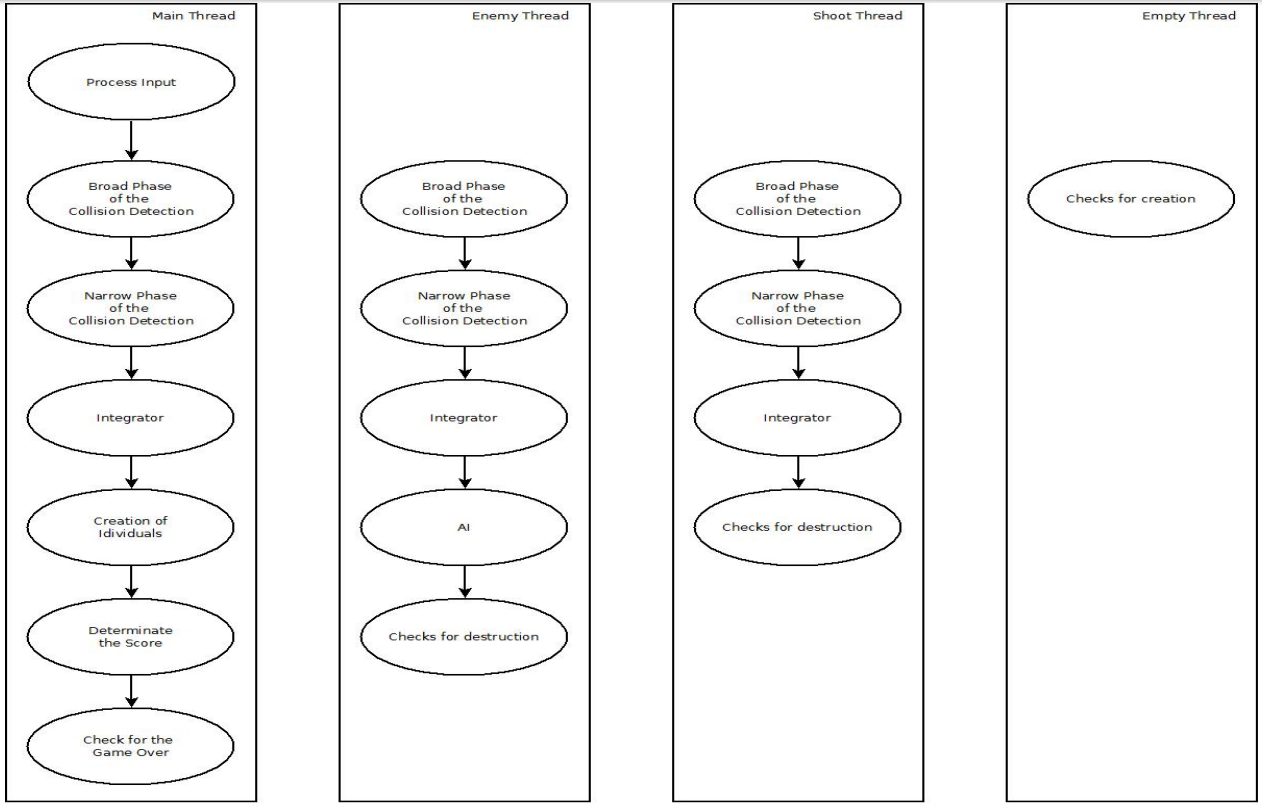


- The GPU is responsible for processing the game logic, like:
  - Process the input;
  - The game physics;
  - The entities/enemies behavior;
- This way we avoid the CPU-GPU data transfers;

Architecture

- The game logic is divided in 4 different threads:
  - Main: which process score and player input;
  - Enemy: which process the enemies behavior;
  - Shoot: which process the shot behavior;
  - Empty: which does not process noting;

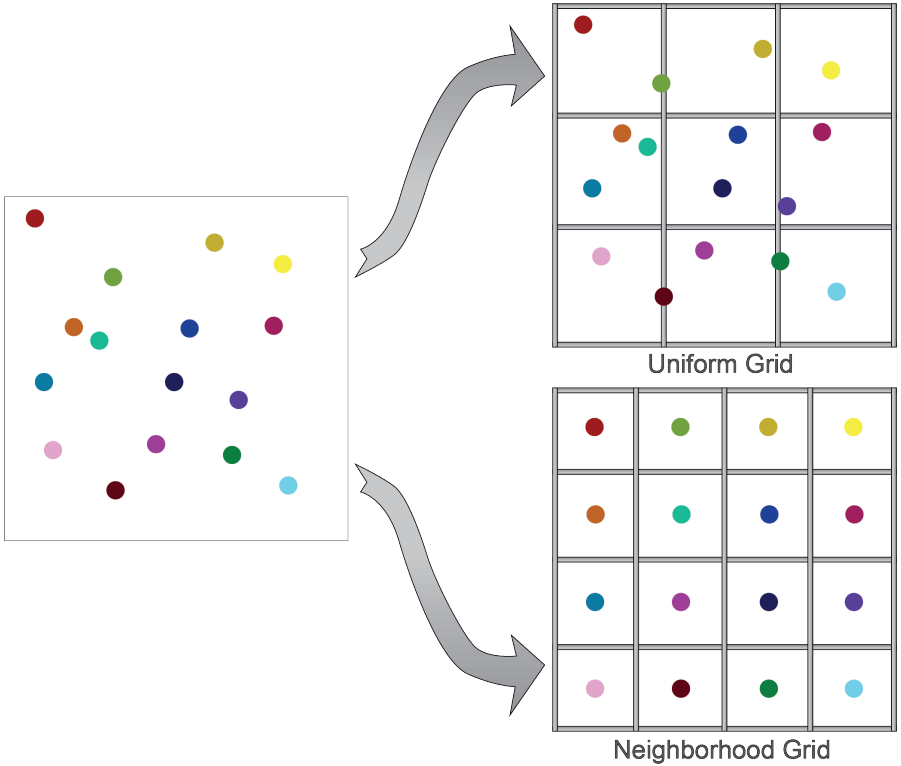
Threads


Main Thread
Process Input
Broad Phase of the Collision Detection
Narrow Phase of the Collision Detection
Integrator
Creation of Idividuals
Determinate the Score
Check for the Game Over
Enemy Thread
Broad Phase of the Collision Detection
Narrow Phase of the Collision Detection
Integrator
AI
Checks for destruction
Shoot Thread
Broad Phase of the Collision Detection
Narrow Phase of the Collision Detection
Integrator
Checks for destruction
Empty Thread
Checks for creation

Broad Phase of the Physics

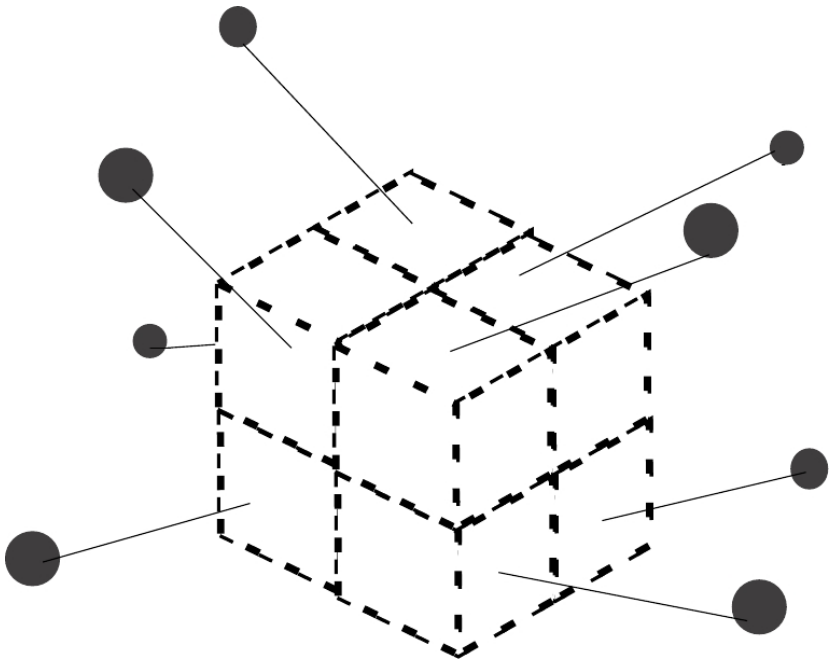
- The same method for neighborhood gathering of the collision detection is used for simulating the enemies “vision”;
- For that our framework can use two kinds of neighborhood gathering algorithms: Uniform Grid and Neighborhood Grid;

Neighborhood Gathering



- The uniform grid is a common way of gather neighbors of entities;
- The neighborhood grid is new data structure for this mechanism;

Neighborhood Gathering

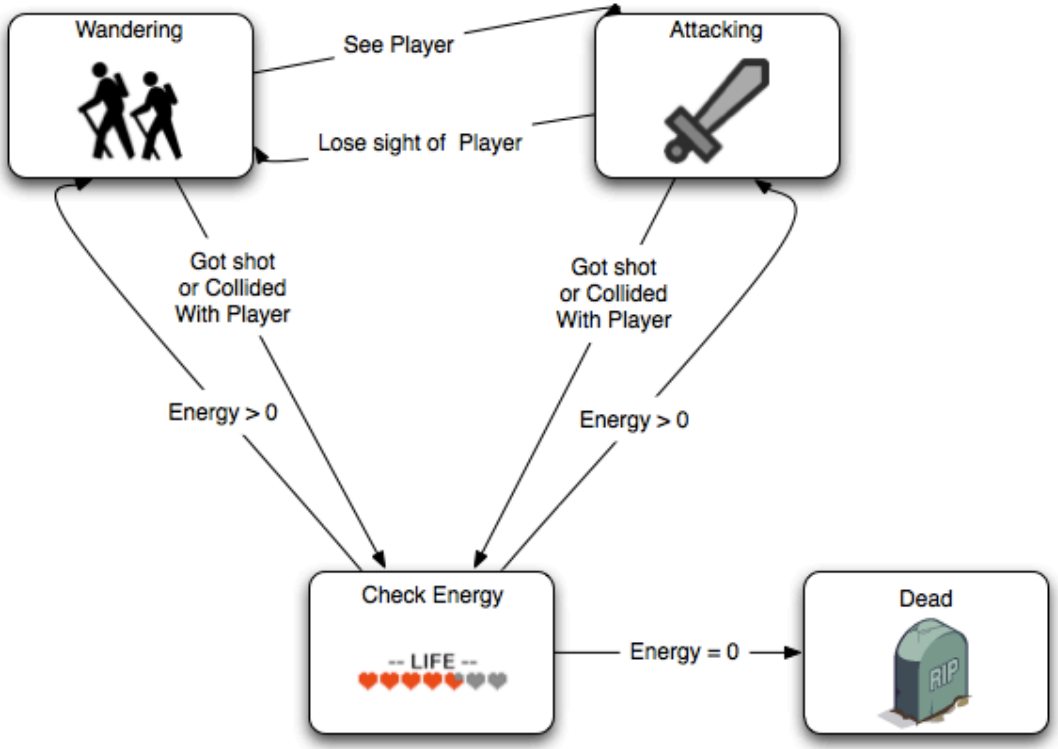


- In the neighborhood grid, each entity is mapped to a individual cell (1:1 mapping) according to its spatial location;
- Particles that are close in a geometric neighborhood sense are mapped to be close in the grid sense;
- In order to keep the neighborhood grid property, a sorting mechanism is done.

AI

- This framework implements state-machines for the AI of the enemies;
- And also behavior of allies and scenery objects and entities;

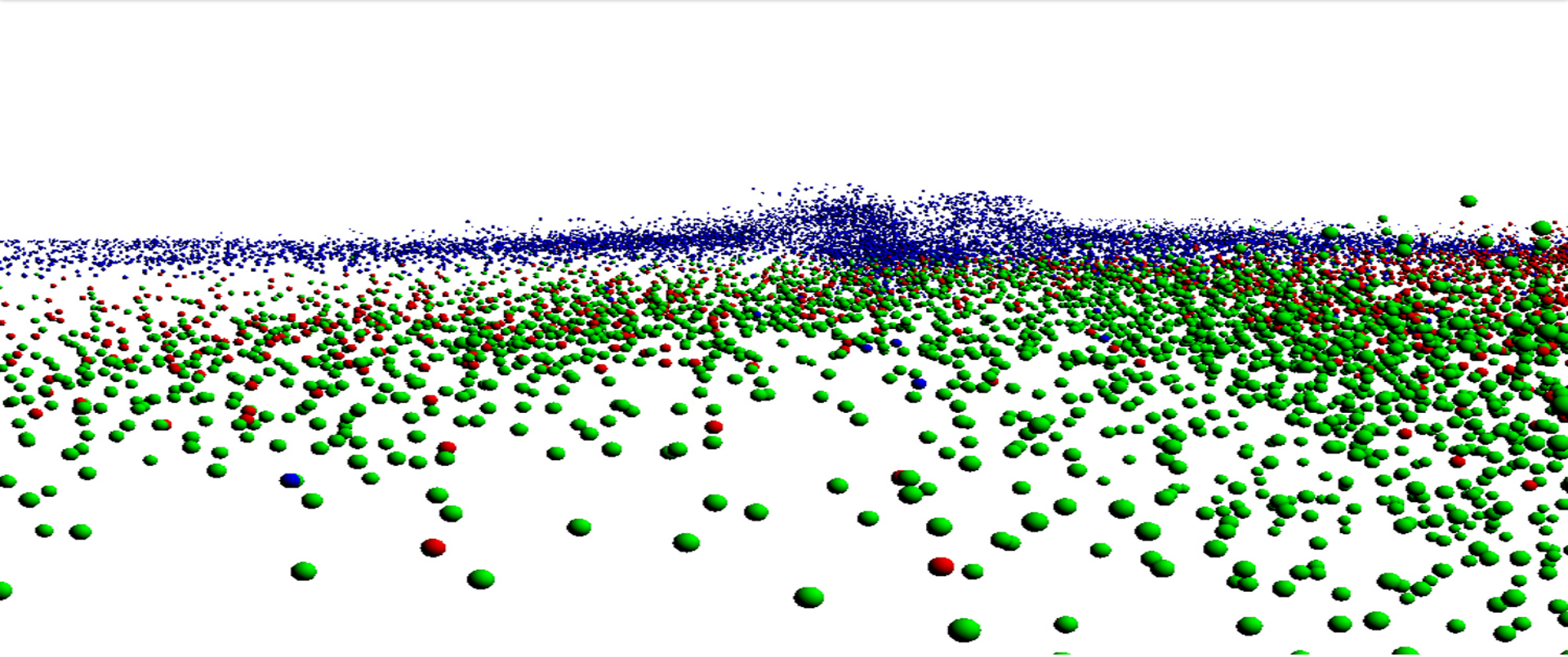
AI: Example


Wandering
See Player
Attacking
Lose sight of Player
Got shot
or Collided
With Player
Got shot
or Collided
With Player
Energy > 0
Energy > 0
Check Energy
-- LIFE --
Energy = 0
Dead
RIP

Test Crowd

- Using the framework we have implemented a flocking boids scenarios;
- Can process and render up to 2MM boids in real-time, while using the CPU can only 10K and a misc. CPU-GPU 100K;

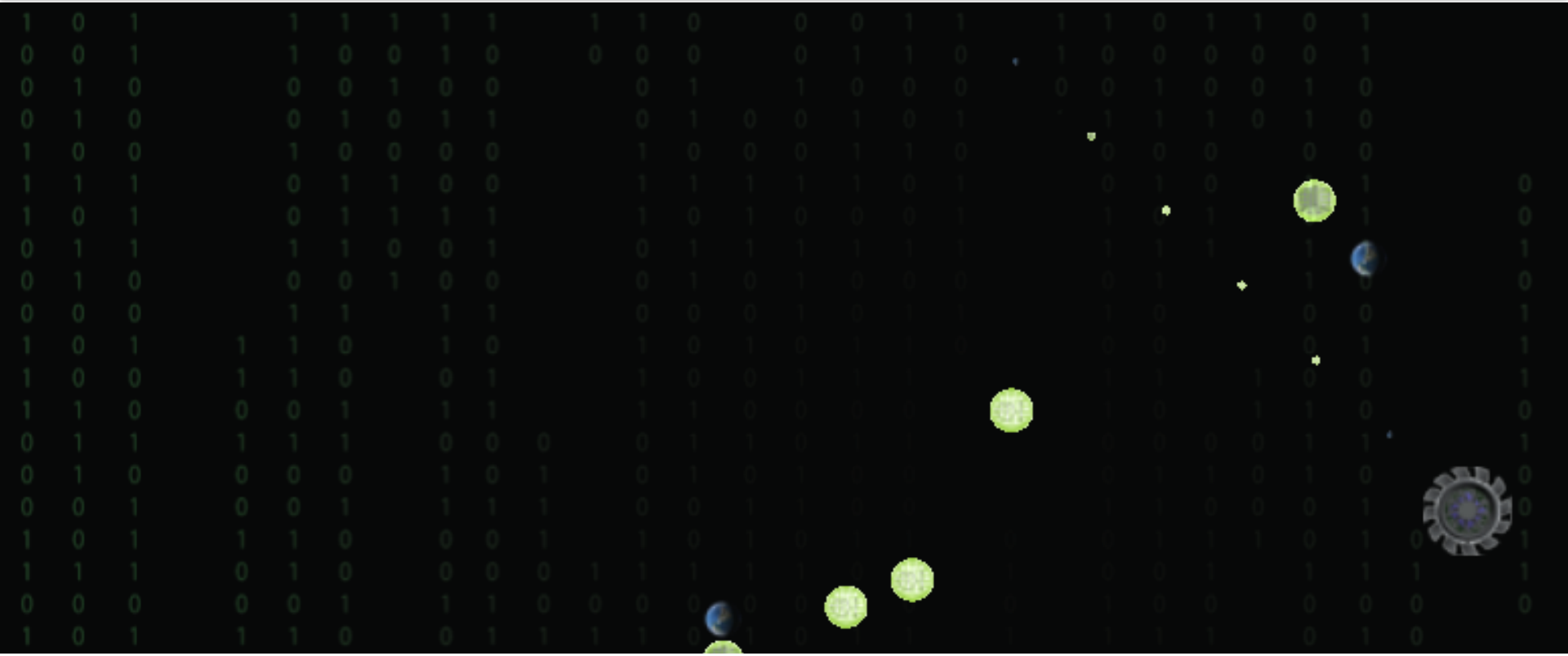
Test: Crowd Test



Test Game: GPU Wars

- Game based on Geometric wars;
  - The player represents a GPU card inside the Computer and needs to process polygons, shaders and data by shooting them;
- Can process and render up to 16K entities in real time while the misc. CPU-GPU can process 8K and CPU 4K;

Test Game: GPU Wars



Conclusions

- We presented a framework for simple games that:
  - uses practically only the GPU to process the logic;
  - Integrates the physics with AI;

Questions

mjoselli@nullpointer.com.br