Boids implementation inside the Blender Game Engine

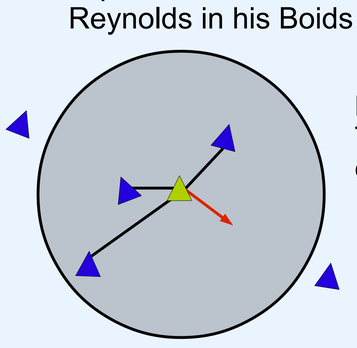
Myrna I. Merced Serrano, Advisor: Gordon Erlebacher
Department of Scientific Computing, Florida State University
mim09c@fsu.edu

Abstract

Blender is a free modeling/simulation software that has been out since 1993, most used only to create 2D and 3D content. It has recently been extended to include modeling, texturing, animation, particle simulation, rendering, game creation, etc. The Blender Game Engine is a very powerful tool, allowing games to be created without the need for explicit programming. Although Blender has extensive particle-based tools, including hair styling, these are absent from the game module. A submodule of the particle system is a rather sophisticated Boid System. In this project we intend to incorporate a Boid system inside the Blender Game Engine, enhancing Blender's capability, leading to many opportunities for AI-based algorithms, including Particle Swarm Optimization, manipulation of crowds, etc. The collective behavior of Boids is called flocking, which can be characterized as an emergent behavior caused by following three steering behaviors: separation, alignment, and cohesion. Boids are commonly used in games as non-player characters since they can behave like real entities by themselves without the need for explicit control. Our implementation involved the development of a new Modifier inside Blender. This modifier is called RTPS because it depends on the library Real-Time Particles System (RTPS) developed by Ian Johnson as part of his work on Smooth Particle Hydrodynamics (SPH). RTPS is a library that currently defines two different particle systems: SPH and FLOCK, which is the system presented here. RTPS also incorporates CPU and GPU implementations of both systems. For the GPU implementation OpenCL was chosen as the GPU programming language to ensure portability between different graphics cards.

Flocking

Flocking is the interaction between the behaviors of entities. This entities are called boids. Flocking can be simulated by the implementation of the three steering behaviors introduced by Craig Reynolds in his Boids model of flocks, herds and schools.



Separation

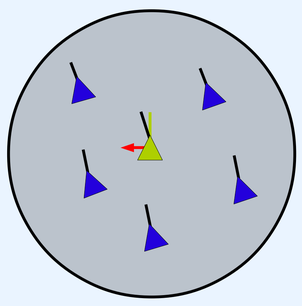
Maintains a minimum distance from each other. This helps to prevent crowding and potential collisions between boids.

$$vSeparation = \frac{1}{M}\sum_{j=1}^{k}\frac{normalize(p_i - \hat{p}_j)}{length(p_i, \hat{p}_j)}$$

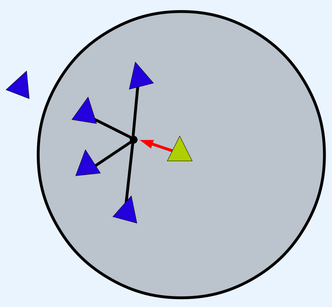
Alignment

Maintains all boids heading to the same direction. Each boid steers towards the average velocity of their local neighbors.

$$vAlignment = \left[\frac{1}{k}\sum_{j=1}^{k} v_j\right] - v_i$$



Cohesion



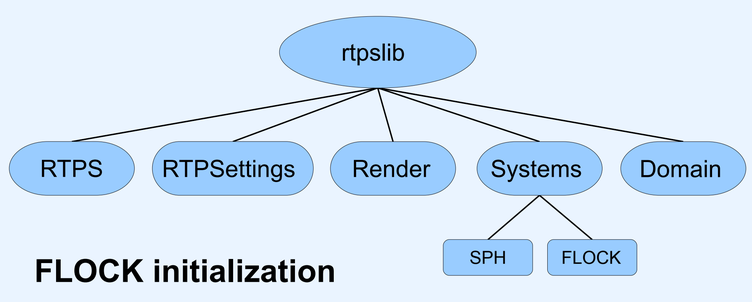
Maintains all boids together as a flock. Each boid steers towards the average position of their local neighbors.

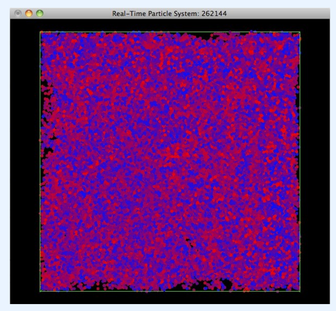
$$vCohesion = \left[\frac{1}{k}\sum_{j=1}^{k} p_j\right] - p_i$$

Algorithm

foreach *Boid i* **do**
  Compute *FindFlockmates*(*i, search_radius*)
  if *flockmates.size*( ) > 0
    Compute *Separation*(*i, flockmates*) → *acc_separation*
    Compute *Alignment*(*i, flockmates*) → *acc_alignment*
    Compute *Cohesion*(*i, flockmates*) → *acc_cohesion*
  end
  Set *vel_separation* = *acc_separation* * *weight_separation*
  Set *vel_alignment* = *acc_alignment* * *weight_alignment*
  Set *vel_cohesion* = *acc_cohesion* * *weight_cohesion*
  Set *acceleration* = *velocity*[*i*] + *vel_separation* + *vel_alignment* + *vel_cohesion*
  if *acceleration.length*( ) > *maximum_speed*
    acceleration = *normalize*(*acceleration*) * *maximum_speed*
  end
  Set *velocity*[*i*] = *v* + *acceleration* **Comment**: *v* is an optional velocity field
  Set *position*[*i*] += *dt* * *velocity*[*i*]
  Compute *CheckBoundaries*(*position*[*i*])
end

Real-Time Particle Systems Library





FLOCK initialization

- Set the maximum number of boids
- Setup the FLOCK settings
- Setup the domain
- Setup the initial conditions for the FLOCK
- Set the renderer

FLOCK update

- The only step of updateCPU is to call the integration method which computes the entire algorithm.
- The steps of updateGPU are: 1) setup the boids for the neighbor search, 2) neighbor search, 3) compute the *steering behaviors*, and 4) compute the final velocity, and update the position.

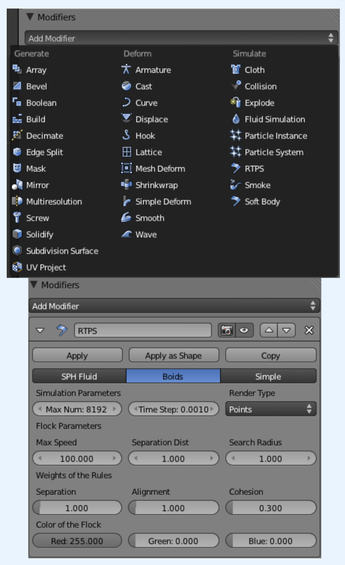
Insert Boids to FLOCK

- Two initial configurations are currently available: box and sphere.
- The dimensions are send to the domain class which is going to fill the vector of the positions.

FLOCK parameters

- Ou implementation has six parameters that can be set by the user: maximum speed, desired separation distance, neighbor search radius, and the weights for the steering behaviors.

RTPS Modifier for Blender Game Engine



The Blender Game Engine was enhanced by adding a new modifier that is able to create and simulate real-time FLOCK and SPH particle systems.

Blender source code modifications

I. Create the connection between RTPS and Blender
- Development inside the Game Engine
- Import RTPS library
- Create and initialize the RTPS object

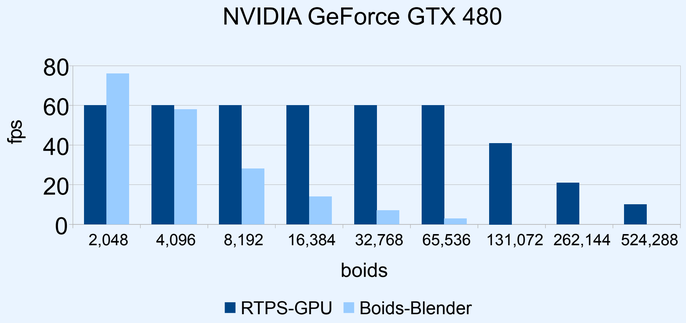
II. Develop the functionality of the RTPS Modifier
- Development inside Blender
- Create a struct with the RTPS settings
- Define and initialize each of the settings

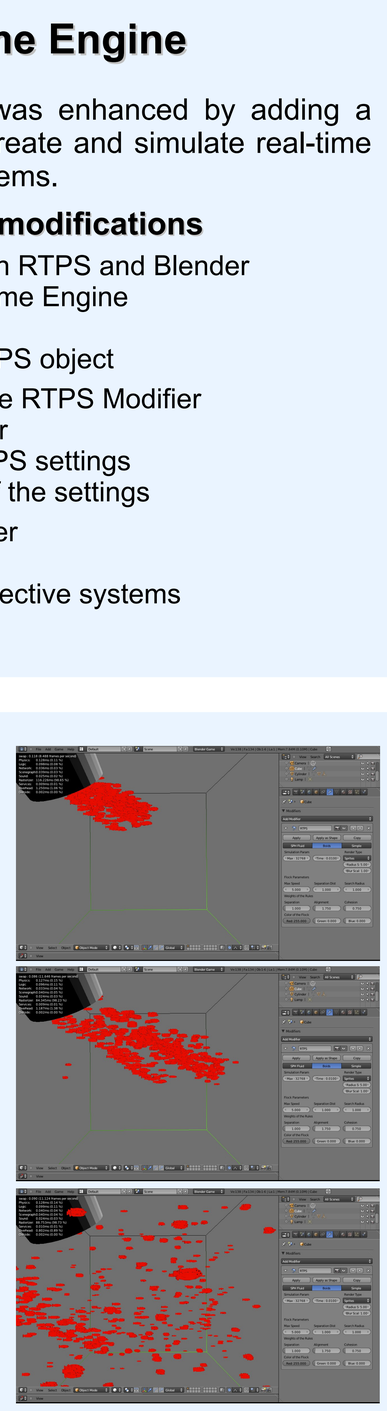
III. Develop the UI for the Modifier
- Development in Python
- Add the settings to the respective systems

Results

The previously described system can be run in the Blender Game Engine successfully.

The performance was measured and RTPS Boid system running in a GTX 480 GPU is clearly more faster than the Boid system already available in Blender.





Conclusions

The Blender Game Engine was enhanced by adding a custom modifier. This custom modifier calls the RTPS library which has all the implementation for the FLOCK and SPH systems. This is a work in program, only simple 3D motion of the boids is presented here. Ideally, the capability of our game engine Boid system should be similar to that already available outside the Blender Game Engine, except for greatly enhanced efficiency, since it runs on the GPU.

References

[1] Craig Reynolds, "Flock Herds and Schools: A Distributed Behavioral Model", SIGGRAPH, 1987.
[2] Craig Reynolds, "Steering Behaviors for Autonomous Characters", Game Developers Conference, 1999.

Acknowledgements

Thanks to Gordon Erlebacher for his advices, thanks to Ian Johnson, Evan Bollig, and Andrew Young for all their help, and thanks to the Department of Scientific Computing.