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 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

References


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.

Flock Herds and Schools: A Distributed Behavioral Model

Craig Reynolds

Cohesion

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

\[ v_{\text{Cohesion}} = \frac{1}{k} \sum_{j=1}^{k} p_j - p_i \]

Alignment

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

\[ v_{\text{Alignment}} = \frac{1}{k} \sum_{j=1}^{k} v_j - v_i \]

Separation

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

\[ v_{\text{Separation}} = \frac{1}{M} \sum_{j=1}^{M} \text{normalize}(p_i - p_j) \]

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.

\begin{align*}
\text{Compute Separation}(\text{flockmates}) & \rightarrow v_{\text{Separation}} \\
\text{Compute Alignment}(\text{flockmates}) & \rightarrow v_{\text{Alignment}} \\
\text{Compute Cohesion}(\text{flockmates}) & \rightarrow v_{\text{Cohesion}}
\end{align*}

Algorithm

\begin{verbatim}
for each Boid bdo
  Compute Flockmates(bdo, search_radius)
  if flockmates.size > 0
    Compute Separation(flockmates) \rightarrow v_{\text{Separation}}
    Compute Alignment(flockmates) \rightarrow v_{\text{Alignment}}
    Compute Cohesion(flockmates) \rightarrow v_{\text{Cohesion}}
  end
  Set vel_{\text{Separation}} = acc_{\text{Separation}} \times weight_{\text{Separation}}
  Set vel_{\text{Alignment}} = acc_{\text{Alignment}} \times weight_{\text{Alignment}}
  Set vel_{\text{Cohesion}} = acc_{\text{Cohesion}} \times weight_{\text{Cohesion}}
  Set acceleration = vel_{\text{Alignment}} + vel_{\text{Cohesion}} + vel_{\text{Separation}}
  if acceleration.length > maximum_speed
    acceleration = normalize(acceleration) \times maximum_speed
  end
  Set velocity(bdo) = v + acceleration \hspace{1cm} \text{Comment: v is an optional velocity field}
  Set position(bdo) = bdo + velocity(bdo)
  Compute CheckBoundary(position(bdo))
end
\end{verbatim}

Develop the UI for the Modifier

I. Create the connection between RTPS and Blender
   - Development inside the Blender
   - 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

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 Blender
   - 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.