Python Boid Crowd System for Houdini

Project Report
TNCG13, SFX - Tricks of the Trade
Linköping University

Lukas Lundberg[*]

June 4th, 2012

Abstract

This paper describes the implementation of the Boid crowd system for the 3d animation software Houdini [1]. The implementation of the methods is written in the programming language Python [2] for Houdini. A Houdini tool often referred to as a Digital Asset with controls is also provided.

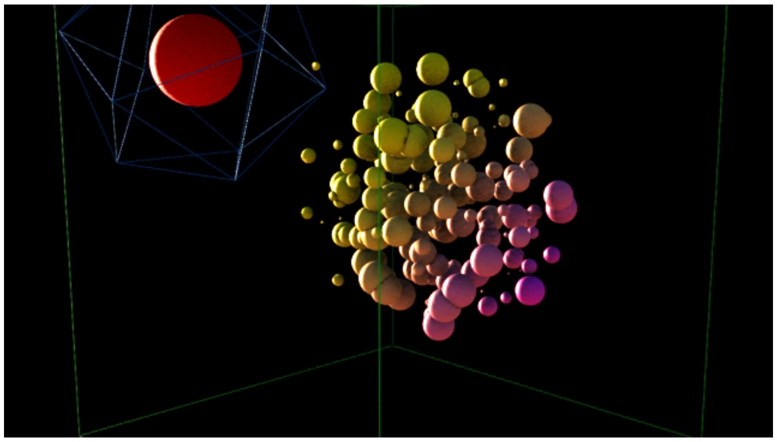


Figure 1: Python Boid Crowd System render image in Houdini

1 Introduction

In a philosophy context the Turing test propositioned that; if a machine acts as intelligently as a human being, then it is as intelligent as a human being. The certain intelligence of a machine is also called artificial intelligence [4]. Crowd simulations are often considered to be intelligent systems and can be seen in many context for example a crowd of enemies behaviour in a video game, scientific reasons to study the motion of animal grouping behaviour, or within visual effects to visualize an epic battle with thousands of warriors for example in the movie Lord of The Rings [3].

A crowd system is populated with units often refereed to as agents. An agent is a decision-making individual with diverse characteristics. The decisions of the agents are based on certain rules set to the crowd system. In visual effects the crowd simulation behaviour have been around some time. One behaviour set for crowds simulation from an artificial life program is called Boids, and was developed by Craig Reynolds in 1986. [5]

[*]e-mail:luklu932@student.liu.se

2 Method

The paper Reynolds [5] describes is an emergent behaviour, meaning that the action of movements of an agent is decided upon the interaction with certain rules between the agents refereed to as Boids.

2.1 Distance

The Euclidean distance between two given points in space is the length of the line segment connecting the points position. The rules of the Boids behaviour are based on the distance between the Boids, as can be seen in the equation below,

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{1}$$

$$d = distance, (x_1, y_1, z_1) = point_1, (x_2, y_2, z_2) = point_2$$

2.2 Behaviour Model

The Boid steering behaviour obey three rules; separation, cohesion and alignment.

The separation rule allows the Boids to avoid collision and steer away from close by Boids. The cohesion rule allows the Boids to steer towards the average position of close by Boids. The alignment rule allows the Boids to steer towards the average heading of close by Boids. The rules are illustrated below with figures taken from [6],

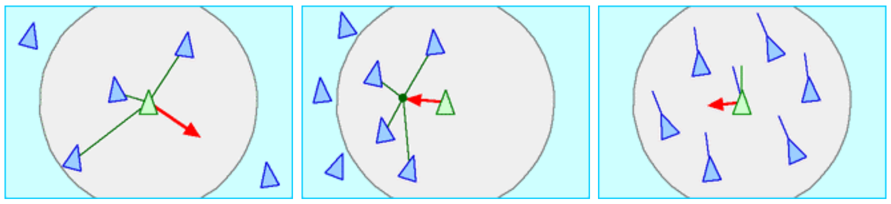


Figure 2: Separation, cohesion and velocity alignment.

3 Implementation

The implementation of the methods was written in the scripting language Python and implemented in Houdini as a Digital Asset with a user interface.

3.1 Overview

For every frame each Boid determine which Boids to be considered to be close by Boids based on it current position and the given neighbourhood radius. Based on the close by Boids the Boid rules are executed and calculates a new velocity and the position for the current Boid.

A container is also defined to keep the Boid in a certain area. If the Boid moves outside the walls of the container the Boid will change the velocity direction.

3.2 Predator and Prey

An additional behaviour to the crowd system was added by flagging one of the Boid agents to be a predator for the others to avoid as prey. The new rule is based on the separation rule, but only to steer away if the prey is in the predator sight.

3.3 Houdini

Houdini is a 3d animation software with an open environment and support of scripting for Python among other API[1] . There are several domains within Houdini which let us access different information and tools. The implementation was made as a Python node in the SOP(surface operator) domain to gain access to geometry and surface operations for procedural modeling and animation. The Python node can easily be modified and instant recompiled without having to install a third package.

The implementation is able to generate a crowd system from a custom polygon geometry model. The output of the system is a set of points with the Boid agents attributes that can be modified with the tools in Houdini, as well with the inbuilt renderer Mantra.

3.3.1 Feedback SOP

For the crowd system to work the Boids location and velocity from the previous frame is needed to make the Boid decision. As default settings Houdini[2] do not store information about the previous frame at the surface level. In other domains as DOPs (Dynamic operator) and POPs (Particle operator) the cache take care of the issue for us but for SOPs where we have our Python code the solution is not straight forward. However there is a node that comes with Houdini but as default is not installed, the feedback SOP node. This node stores feedback information from the previous frame and make it possible to fetch the Boid attributes as the position, velocity and more.[3]

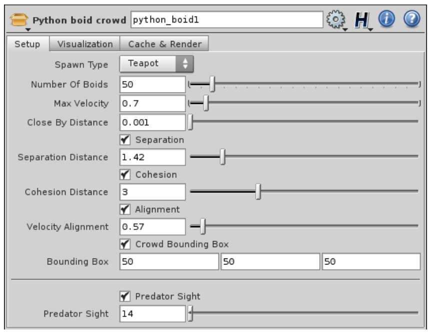
3.3.2 Visualization

The prey Boids can be visualized with colors by the length of the velocity vector and the distance to the predator position. As models the Boids can be visualized as spheres with random sizes generated by the point number and the predator attribute. Or the prey pattern can be visualized as connecting lines between the prey.

3.3.3 Digital Asset

Custom operator types built from node networks is often refereed to as Digital Assets in Houdini. The network is encapsulated with custom parameters or handles at top level to control the asset. The digital asset let us package and share our network with others.

The implementation is provided as a Digital Asset with a user interface for the user to modify the crowd system. Given controls for setting up the Boid crowd system as spawn type, number of Boid agents, maximum velocity, toggle the Boid rules on and off and more. Optional visualization with colors and model. The Cache & Render tab let you store the simulation to cache or as a geometry file sequence for post work. See figure below,



Python boid crowd	python_boid1
Spawn Type	Teapot
Number Of Boids	50
Max Velocity	0.7
Close By Distance	0.001
	✓ Separation
Separation Distance	1.42
	✓ Cohesion
Cohesion Distance	3
	✓ Alignment
Velocity Alignment	0.57
	✓ Crowd Bounding Box
Bounding Box	50 50 50
	✓ Predator Sight
Predator Sight	14

Figure 3: The implemented Digital Asset with custom parameters and handles.

[1]Application Programming Interface

[2]Houdini Master Version 12.0.543.9

[3]Install the Feedback SOP by open up the Houdini shell terminal, type *proto_install* and choose the node. After the installation is complete restart Houdini.

4 Results

The rendered image results from the implementation can be seen below,

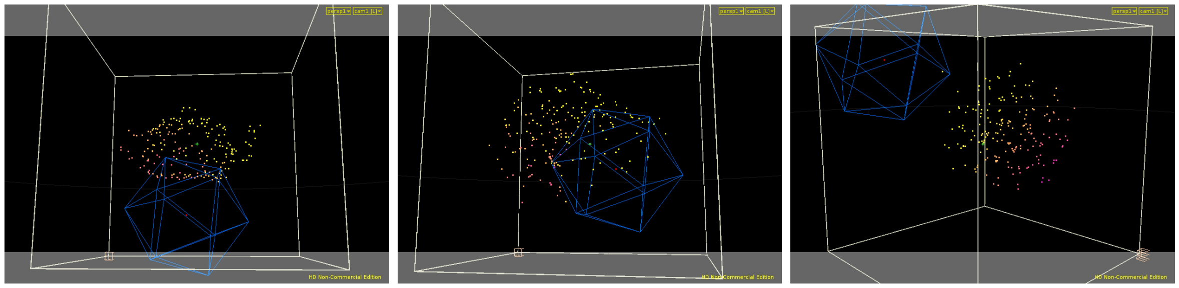


Figure 4: Houdini Viewport render

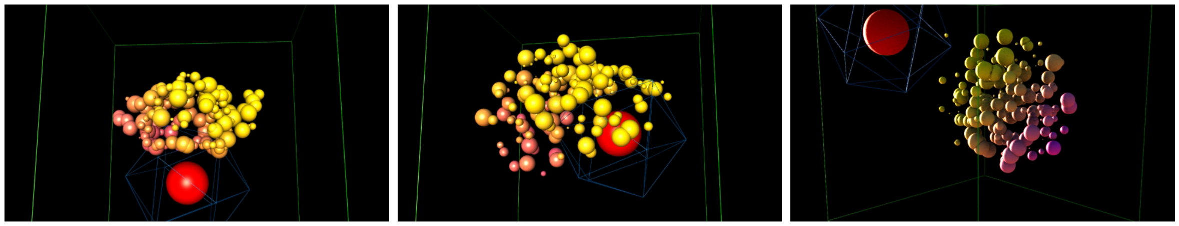


Figure 5: Houdini Mantra Micropolygon render

5 Conclusion

The project primarily demonstrations the flexibility a Python script with Houdini can generate. The implementation with Python may not be the fastest scripting language to approach a heavy crowd simulation. However as Python is one of Houdini main scripting language it is easy to access and modify, and can be a worthy tool to shape your blue print.

The Python script could be extended with more rules and conditions, for example avoiding obstacles and goal seeking. In Houdini the network could be heavily extended for example with detailed models, animation cycles, custom shaders and more controls.

References

[1] Side Effect Software Inc. http://www.sidefx.com/

[2] Python Programming Language. http://www.python.org/

[3] Lord of the Rings. Film. Directed by Peter Jackson. USA New Line Cinema(2001)

[4] Alan Turing. *Computer Machinery and Intelligence* (http://loebner.net/Prizef/TuringArticle.html)(1950)

[5] Craig Reynolds.*Flocks, herds and schools: A distributed behavioral model* (SIGGRAPH '87)(1987)

[6] Boids. http://www.red3d.com/cwr/Boids/