UNITY3D PLUGIN
FOR SIMULATING CROWDS

Client
dr. Roland Geraerts (r.j.geraerts@uu.nl)
Virtual Worlds
Dept. of Information and computing sciences
Utrecht University (version 23-01-2017)
A huge challenge is to simulate tens of thousands of agents (e.g. characters, robots, pedestrians, or monsters) in real-time where they realistically avoid collisions with each other and with obstacles present in their environment, especially near narrow passages where cooperative behavior is required. This environment is usually three-dimensional, e.g. it may contain bridges where agents can walk over and under as well. The environment can also dynamically change, e.g. a bridge may partially collapse.

We study how we can automatically create a data structure that represents the walkable surfaces in virtual environments, and how this structure can be updated dynamically and efficiently when the environment changes. We refer to this structure as a navigation mesh, which is a decomposition of the walkable surfaces into simple, non-overlapping shapes. See Figure 1 (right) for an example. This mesh enables efficient crowd simulation, which is our topic of research. We study and develop a crowd simulation framework and its components, which ranges from global (AI) planning to local animation. We create models for realistic crowd behaviors, which includes studying how (groups of) agents move and avoid collisions in such environments.

We continuously integrate our research into our UU Crowd simulation (UUCS) software which is used by many research institutes and companies. We run simulations in realistic environments and game levels to study the effectiveness of our methods.

**Simulation framework**

To allow for efficient and flexible crowd simulations, our simulation framework consists of the following five levels of planning:

---

1 Crowd simulation members of the Virtual Worlds group (Computer Science, Utrecht University) are Roland Geraerts, Wouter van Toll, Arne Hillebrand and Angelos Kremyzas.
• **High-level planning:** Agents determine their global actions, and each action results in a geometric path planning query (i.e. “move from point A to point B”). In this phase, we support for instance social groups, large groups, re-planning of agents when the navigation mesh changes or when crowd densities change.

• **Global route planning level:** Given the start and goal position of an agent (or a group of agents), we compute a so-called indicative route that the agent (or group) should globally follow during the simulation. Our navigation mesh can generate various types of indicative routes, such as short paths with some amount of minimum distance to the obstacles, or paths that avoid areas with a high crowd density. Indicative routes can also be optimized for camera movement or stealth movement.

• **Route following:** In each simulation step, the agents compute an attraction point on their current indicative route, and they compute their preferred velocity (i.e. their preferred walking speed and direction). We have also developed a route following method that allows agents to have personal preferences for different terrain types.

• **Local movement:** In each simulation step, each agent converts its preferred velocity to an actual velocity that adheres local rules. These rules include resolving collisions with other (moving) agents, avoiding possible future collisions, adapting the velocity to the local crowd flow, and maintaining coherent social groups. When all velocities have been computed, the agents apply their velocity for one simulation step, and the next step begins.

• **Animation:** In many applications, the crowd should be visualized smoothly in real-time. In our model, this task is delegated to Unity3D which offers sophisticated 3D animation and rendering.

More information about the framework can be found in the following paper.

**API and Unity3D Plugin**

We have compiled the UUCS software as a DLL plugin that can be linked to other programs, such as the Unity game engine. Unity can communicate to this DLL via an API that exposes the most important UUCS functionalities (e.g. “add a character at position X”, “let character Y plan a path to position Z”, and so on).

We have created an initial version of a Unity3D plugin, see the figure below / on the next page. It’s not ready though for mass consumption yet.

---

**Purpose**

The goal of the project is to add several components to our basic Unity3D crowd simulation plugin based on our API. The plugin has to work for 3D environments in which the traversable pieces in the environment can all be seen from above (i.e. these pieces don’t overlap when projected on the ground plane). All of the software pieces need to be well-designed, implemented and tested. At the end of the project, the prototype, demos and tutorials need to be ready to be submitted to the Unity asset store. The targeted users of the software are game developers, students and researchers.

---

3 Our software can handle multi-layered environments though, such as a train station with multiple connected levels, but making a plugin for such environments might ask too much of a single software project.
**What’s the functionality of the game/ and/or software?**

To get familiar with our software, you need to create some tutorials and demo’s. Next, you need to work on a few basic components that will improve the current plugin, and to create two editors: one that allows the user to better model the 3D walkable/non-walkable surfaces, and one that eases setting up a crowd scenario. Finally, you need to create materials for PR purposes, see the subsections below.

**Getting familiar, or: Tutorials and demos**

A simulation of a crowd can be difficult to set up, because (1) a valid description of the traversable surfaces needs to be given, and (2) setting up a desired scenario involves making many choices. Therefore, we want to help the **end user** by providing him/her a set of tutorials and demo’s to decrease the learning curve of using our software and to teach them about the software’s possibilities.

Please work on the following requirements for **version 1.0**:

- Make one in-depth tutorial that explains how to model the walkable surfaces in the environment, and how to deal with elevated terrains.
- Make two tutorials, where each one addresses one of the following important features. Choose from: dealing with dynamic obstacles, social groups, changing character parameters during runtime, and adding a crowd to your game.
- Make two interesting demo projects, where each one addresses one of the following pain points usual game developers have. Choose from:
  - how to model the walkable space in Unity3D, given an imported 3D scene file (like .OBJ);
  - how to create a ‘believable/interesting’ crowd;
  - how to handle dynamic obstacles;
  - how to set up and include social groups;
how to handle dense crowds (e.g. using density-based planning, left/right indicative route, streams);
how to let the crowd interact with a playable character.

**Basic Required Components**

The following components/features need to be developed:

- Expose the unexposed API features to the user.
- Expose all relevant errors in a user-friendly way to the user.
- Extend Unity3D’s editor to interface with our API. You need to apply well-established Unity3D design principles to provide high level interfaces to our API. So the usability and user-friendliness are important factors. There are two main components which need their own GUIs:
  - Modelling the walkable environment.
  - Setting up the simulation.
- Improve the way how we convert geometric primitives (such as cubes and spheres) to projected non-traversable polygons. Now, we simply take each face (whose normal points up), and project it onto the plane. You need to remove the (parts of) faces that lie beneath the walkable surface.
- Find out which precautions need to be taken to ensure that the software is not misused.

**Version 2.0: Editors**

When all the above functionality has been implemented, please continue with version 2.0.

One way of modelling the walkable surfaces is to call API functions/scripts that create obstacles. Users obtain more freedom when they can edit the traversable/non-traversable space themselves in an editor (inside Unity3D).

- Extend the Unity3D editor to model the 2D polygonal traversable/non-traversable surfaces. For this editor, providing an orthographic top view suffices. Please include at least the following options:
  - Adding and removing traversable and non-traversable surfaces;
  - Changing the shape of the surfaces;
  - Vertex snapping;
  - Undo/redo functionality;
  - Loading/saving mechanisms;
  - Projecting the walkable surfaces onto the 3D geometry.

Next, instead of using API calls to set up a crowd, create a GUI that eases setting up a crowd scenario:

- Extend the Unity3D editor to set up a crowd scenario. For instance, in the editor, allow the user to define
  - spawn and goal areas;
- custom indicative routes;
- social groups;
- sub-goals for agents and groups;
- character profiles.

- Add functionality to load and save a scenario.

And finally:

- Write a (well formatted) document about the design of the software you have created.
- Improve/extend our tutorial that creates a basic crowd simulation using the C#-wrapper around the API, and the two new editors. The tutorial should describe all steps required from setting up the software to running the first basic crowd simulation in a simple environment with a few obstacles.

**PR**

Please prepare the following items for PR-reasons (NB items should be put on an internal webpage, targeted at the end user) which is not visible for the outside world):

- Put the tutorials and demos on a well-formatted web page;
- Create a visually-appealing project with many features to impress people;
- Create a professional video that will be used for marketing the plugin on YouTube. The video needs to be of high-quality (contents, low compression, at least 1080p).

**APPLICATION AREA**

The application area is crowd simulation.

The plugin is used in research and education by many research institutes, including ours. In our department, the plugin is used in the course Path planning⁴. See this overview⁵ for projects we’ve been doing using the crowd simulation software.

The plugin can be used in many commercial areas, including the serious and entertainment game-industry (e.g. crowds in applications used for training or in architectural applications, or in computer games), the safety and security market (e.g. to perform evacuation studies in big infrastructures, to study crowd flow optimizations in cities, or to real-time predict agents’ motions during an event), and simulations in the intelligent transport sector. There are plans for an academic startup, and the results of this project can be used here (if our university provides a license for that.)

---

⁴ See [http://www.cs.uu.nl/docs/vakken/mpap/#assignment2](http://www.cs.uu.nl/docs/vakken/mpap/#assignment2) for the course description.

⁵ See [http://www.staff.science.uu.nl/~gerae101/UU_crowd_simulation_outreach.html](http://www.staff.science.uu.nl/~gerae101/UU_crowd_simulation_outreach.html) for a list of outreach activities.
Unity3D uses its internal path planning engine that is based on the open source project Recast/Detour created by Mononen, see http://masagroup.github.io/recastdetour/index.html and https://github.com/masagroup/recastdetour for the current distribution.

You should also do some research yourself, i.e. make a list of existing related plugins, and find out what kind of interfaces they use. Please also look at other path planning/finding plugins, including http://arongranberg.com/astar and http://apexgametools.com for inspiration purposes.

The goal of the research is to find out what the quality of these plugins are, how they look and feel, and how they are designed.

Why is the project interesting?

Crowd simulation is an important AI component in many games and simulations, but we think that its current state of the art can be dramatically improved. It is not only a challenging topic, but there also appears to be many errors and design mistakes in existing algorithms. Consequently, NPCs get stuck, walk through each other, or react in an undesired way on dynamic changes in the environment (like a parked car). Also, some algorithms are too slow when many NPCs are presented simultaneously. See for example http://www.cs.uu.nl/~roland/pdf/PoorPathPlanning.wmv or http://www.youtube.com/watch?v=izpgMnu_lAk&feature=plcp for a visual impression.

We improve upon a few elements upon the state-of-the art, see http://www.staff.science.uu.nl/~gerae101/UU_crowd_simulation_software.html for a list.

By creating a user-friendly and high-quality plugin, we allow developers to facilitate high-quality, real-time crowds in less time. It will improve the gameplay and immersion in games, and will bring a more realistic crowd simulation solution to developers of other application areas. Our engine is already used for some projects, including crowd flow optimization for the Tour de France in Utrecht and evacuation studies in the Noord/Zuidlijn in Amsterdam. The software contributes to the safety in such areas. From a commercial perspective, we hope we can generate more revenue to finance R&D.

Why is the project interesting for the student?

Students learn to deal with and develop advanced (game)technology, i.e. crowd simulation, navigation meshes, AI, game engines, editors, and plugins. Crowd simulation in games and simulations can still be improved a lot, and a big improvement in this field shows that you have much to offer. So this project is a nice one to have on your CV/portfolio. Optionally, we can pay a visit to a company that uses the software.
DELIVERABLES

You need to deliver the following items:

- The items in the section `What’s the functionality ...`;
- A document with the description and choices of the new plugin;
- Unit tests. For instance, you could use [https://code.google.com/p/googletest](https://code.google.com/p/googletest).
- All software, sources, assets etc need to be put on the UU GIT server and on a USB-stick.

DURATION PROJECT

The project takes place during period 3 and 4 (week 6 through 26, so 6-2-2017 through 30-6-2017).

DESIGN BOUNDARIES

In general, we find light-weight, beautiful, extendable, maintainable and efficient code very important. NB the code you provide will be directly exposed to the end user. More specifically:

- Unity3D is heavily component-based. So the design of the wrapper classes should follow the component-based paradigm of MonoBehaviour scripts. Unity’s own NavMeshAgent\(^6\) \(^7\) and NavMeshObstacle\(^8\) \(^9\) are examples that follow this design paradigm.
- Don’t change our API headers/documentation. You’re highly encouraged to suggest changes though.

We find it always more important to have fewer features that fully work and are of high quality, than to have more features that are incomplete. So please finish a certain feature completely before you move on to the next one. NB Obviously, you can do things in parallel.

Work in many iterations. While your project is running, our engine is periodically improved and a few features may be added.

LIMITING CONDITIONS

Here are some limiting conditions:

- The software needs to be run on an installation of the 64-bit version of the Unity editor, version 5.4.1 or higher.
- The programming language is C#.

---

\(^6\) [https://docs.unity3d.com/Manual/class-NavMeshAgent.html](https://docs.unity3d.com/Manual/class-NavMeshAgent.html)
\(^7\) [https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html](https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html)
\(^8\) [https://docs.unity3d.com/Manual/class-NavMeshObstacle.html](https://docs.unity3d.com/Manual/class-NavMeshObstacle.html)
\(^9\) [https://docs.unity3d.com/ScriptReference/AI.NavMeshObstacle.html](https://docs.unity3d.com/ScriptReference/AI.NavMeshObstacle.html)
• The plugin and all code needs to run on a Window computer. (In a later stage, we will support our plugin to other platforms.)
• The development needs to be done on the UU GIT server. Access can be arranged by sending your list of UU-email addresses to your client.

**SYSTEM INPUT**

The input is an empty Unity3D project, or a file that describes the 3D geometry or 2D walkable/non-walkable surfaces.

**DESIRED BEHAVIOUR**

The user should have a very positive experience when he/she uses the software, documentation, demo projects and tutorials. We hope that developers are positively surprised and impressed by the new possibilities and quality of the demos.

**UNDESIRABLE BEHAVIOUR**

The user shouldn’t feel frustrated when he/she uses the software.

**USAGE**

The number of clicks to get something done should be minimized. The GUI should be consistent.

**VISUALISATION**

The user should also see a convenient and clean GUI. The demo should represent the next generation games / simulations.

**ARTWORK**

There is a high need for graphically educated people. They need to work on a movie and 3D assets that are needed for the movie.

**PLAYER’S/USER’S PERSPECTIVE**

Not applicable.
**ACTIONS / OPERATIONS**

The user should be able to automatically and manually create the walkable/non-walkable surfaces. He/She should be able to set up a crowd simulation quickly.

**REQUIRED MATERIALS**

As mentioned before, the code needs to be written in C#, and needs to run on a Windows computer.

**TEST ENVIRONMENT**

The final code and projects needs to run without compilation errors and warning on the client’s computer (Windows 7, 64 bit).

**CONTACT PERSON**

The crowd simulation research is led by Roland Geraerts, see [http://www.staff.science.uu.nl/~gerae101](http://www.staff.science.uu.nl/~gerae101). He is your client, and you have to report to him. He’s working with some people who are also important for your project, because they have co-developed the software and plugin. See the table below for more information.

<table>
<thead>
<tr>
<th>Name</th>
<th>E-mail</th>
<th>Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roland Geraerts (Assistant professor)</td>
<td><a href="mailto:R.J.Geraerts@uu.nl">R.J.Geraerts@uu.nl</a></td>
<td>Crowd simulation and game technology</td>
</tr>
<tr>
<td>Wouter van Toll (Scientific programmer)</td>
<td><a href="mailto:wvantoll@gmail.com">wvantoll@gmail.com</a></td>
<td>Multi-layered navigation mesh Crowd Simulation Engine Unity3D plugin</td>
</tr>
<tr>
<td>Arne Hillebrand (PhD student)</td>
<td><a href="mailto:a.hillebrand@uu.nl">a.hillebrand@uu.nl</a></td>
<td>Crowd simulation pipeline</td>
</tr>
<tr>
<td>Angelos Kremyzas (Scientific programmer)</td>
<td><a href="mailto:angelos.kremyzas@gmail.com">angelos.kremyzas@gmail.com</a></td>
<td>Crowd Simulation Engine Unity3D plugin</td>
</tr>
</tbody>
</table>