International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8, Issue-1, May 2019

Development of Pedestrian Artificial Intelligence Utilizing unreal Engine 4 Graphic Engine

Fares Abu-Abed, Alexey Khabarov

Abstract*: *Paper demonstrates the implementation of AI for pedestrian simulation in the driving simulator by means of Unreal Engine 4 graphic engine and VR-technologies. The authors review the operation principles of behavior trees and their components. The paper describes the internal structure of a pedestrian AI class and methods of implementing object detection in the field of view and audibility utilizing Unreal Engine 4. The authors give the example of using Environment Query System included in the engine AI system and display the result of executing its queries, which used in the process of the virtual pedestrian behavior tree simulation. Since in the developed driving simulator it is necessary to achieve a high frequency of frame changes and low demands on the resources of the computer system, the article suggests an optimal solution for simulating a large number of pedestrians in a virtual city. The article shows the behavior tree and represents operation algorithms of the pedestrian AI's basic components in block diagrams with annotations.

Index Terms*: *Unreal Engine 4, Environment Query System, behavior tree, artificial intelligence, driving simulator

I. INTRODUCTION

In modern cities of Russia, the traffic intensity and accidents on the roads have increased significantly; consequently, the drivers are subjected to high requirements. This complicates the process of training in driving schools in Russia and adds the need for a significant increase in hours of practical training. To give an initial level of practice to novice drivers and give them an idea of driving a real car, driving schools use driving simulators. Thus, the training of novice drivers with the help of simulators teaches to use the vehicle's controls correctly, excluding the possibility of damage to the vehicle or hurting other traffic participants.

II. DATA

This paper will consider a practical example of using Unreal Engine 4 game engine's (UE4) tools to implement artificial intelligence for pedestrian simulation in the driving simulator. The project is a driving simulator using virtual reality (VR) technologies. A pedestrian AI is necessary for reproducing random situations that can occur when you drive a car in an urban environment.

The goal of the development is to implement a plausible pedestrian behavior model using behavior trees. The pedestrian AI should work independently from the moment of its appearance in the virtual world and automatically react to the changes around it while taking as little computational resources as possible.

III. PEDESTRIAN AI CLASS

Before we begin to talk about the implementation of individual parts of the AI, it is necessary to explain the internal arrangement of the AI system and its operation principles in UE4. The selected game engine uses the component programming approach, as well as a large number of tools available out of the box. The engine is free and has a shared source code, which makes it possible to modify its components, as you like and be independent from developers' update timeline. The development language in UE4 is C ++. In addition, UE4 has its own tool to implement algorithms by means of Blueprints visual scripting system [1]. This is a node-based visual programming language for creating gameplay elements inside Unreal Editor. Like many common scripting languages, it is used to define object-oriented classes or objects in the game engine.

Figure 1 shows a diagram of the main components used in the pedestrian class development.

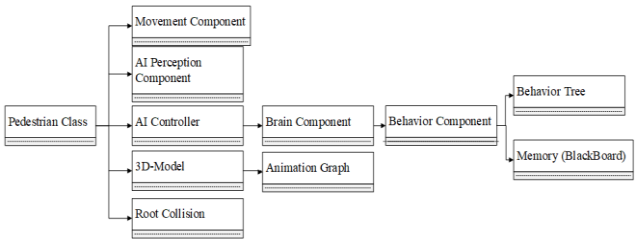

Fig. 1. The structure of pedestrian AI class.

The movement component handles the movement in the three-dimensional space inside the navigation zones. The root collision is used to determine the routes to bypass obstacles or other pedestrians. The movement component handles the movement of the AI on the ground and corrects the route in real time. The root collision is also used to determine hit or overlap events with objects. The AI controller applies the behavior logic to the AI class.

Revised Manuscript Received on 30 May 2019.
***** Correspondence Author
Fares Abu-Abed*, Faculty of International Academic Cooperation / Tver State Technical University, Tver, Russian Federation.
Alexey Khabarov, Faculty of Information Technologies/ Tver State Technical University, Tver, Russian Federation.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

The brain component inside the AI controller contains the behavior tree [2] that describes the AI operation logic.
The Blackboard [3] is responsible for a memory. It is a board of variables used in the operation process of the behavior tree.
AI perception component [4] is responsible for "sight and hearing". It is the basis of the danger detection logic that makes pedestrians run away from an approaching or honking car.
The project also includes classes of crossings and traffic lights. Their function is simple and comes down to forbidding or allowing crossing a road for AIs located in the zone of their influence.

IV. EXPERIMENTAL DESIGN, MATERIALS AND METHODS

We skip the animation graph setup and proceed to the behavior logic coding. It is necessary to implement handlers for events of vehicle detection in the field of view and audibility, hit events with a car and overlap events with a traffic light zone.
The hit and the overlap with the objects are determined by the root collision in the AI class. UE4 generates events when defines an overlap or a hit, in which it transmits the object involved in this event. The AI that received the hit from the car stops working, and its 3D model begins the process of simulating the physical body (Ragdoll) being cast from the car to the ground. The built-in physic engine calculates the physics simulation.
The root collision processes overlap events with the objects in order to determine the AI location near a traffic light. When the AI enters a traffic light zone, the root collision records it to the board of valuables (Blackboard) in the behavior tree. When the AI leaves the traffic light zone, the value is erased.
One of the development tasks is the implementation of a danger detection for the AI. A pedestrian should try to avoid being hit by a car. Here we use the AI perception component to detect objects around the AI.
The AI monitors its radius to detect honks and fast-moving cars and records them to the Blackboard. In 3 seconds, the AI erases this data and returns to normal operation mode.

V. BEHAVIOR TREES

An AI in Unreal Engine 4 is created using behavior trees. The Blackboard stores the data necessary for their operation. Each tree node gives the result of execution to the higher nodes on the branch. The branches are executed from the left to the right. The more to the right the branch is, the lower its priority for execution.
There are 2 primary decision making nodes available for the behavior trees in UE4:
Selector node [5] that will run through its children, from left to right, until one of them succeeds, at which point it will fail back up the tree.
Sequence node [5] that will run through its children from left to right until one of them fails, at which point it will fail back up the tree.
The tasks nodes in the UE4 behavior trees have 3 states:
In progress.
Completed with error.
Completed successfully.
Decorator nodes [6], also known as conditionals in other behavior tree systems, are attached to either a composite or a task node and define whether or not a branch in the tree, or a single node, can be executed. Decorators in UE4 can interrupt the execution of the branch, to which they are attached or the execution of the branches with a lower priority when the execution result is changed.
The behavior trees in UE4 also have service nodes [7] that attach to composite nodes and continue the execution at their defined frequency as long as their branch is being executed. These are often used to make checks and to update the Blackboard. These take the place of traditional Parallel nodes in other behavior tree systems.
An Environment Query system (EQS) [8] is a feature of the AI system in UE4 that is actually made up of a number of different pieces. You must call an Environment Query from a behavior tree, and then the actual Environment Query will use its Generator, reference its Contexts and use its Tests to give the Behavior Tree the highest weighted result.
When the call has been made, a set of elements is generated from the point where the calling object is located using one of the methods proposed by the system in the specified radius. Tests are added to the Generator to select one of these points. They can work as data filters or affect the ratio of conformity to the Test conditions. In addition, they can work simultaneously in these two modes. The result of the execution is one element that matches the given conditions in the best way.
Figure 2 shows the behavior tree of the pedestrian AI.

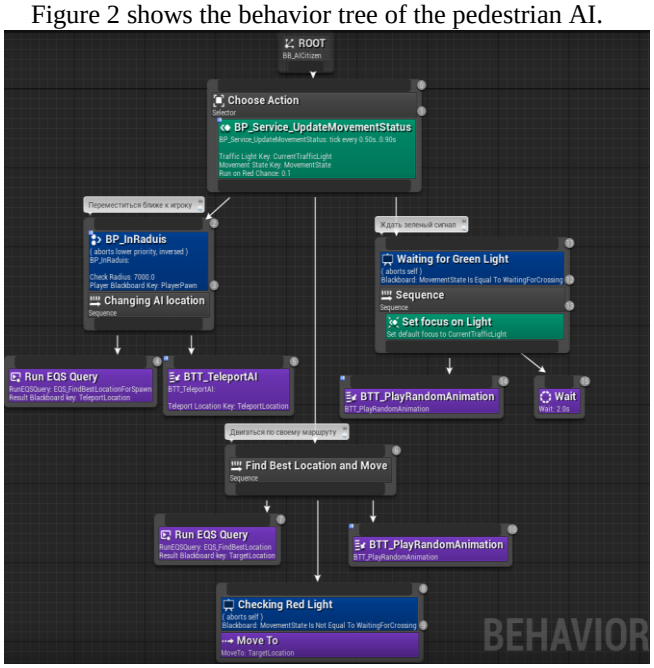

Fig. 2. The behavior tree of the pedestrian AI.

The first node after the root is the selector on which the service is located. The task of this node is to select one of the three sets of an AI behavior with parallel control over the AI movement at the crossroads and a roadway. The service uses keys from the value board.

These keys hold the records of the currently monitored traffic light and the AI's moving mode. During the service operation, the data on the value board is updated to stop the AI when the traffic light is red and avoid danger. The service also accepts the probability parameter of crossing the road when the light is red, which cancels the logic of waiting for the green light for a particular AI. Figure 3 shows the operation algorithm of the service.

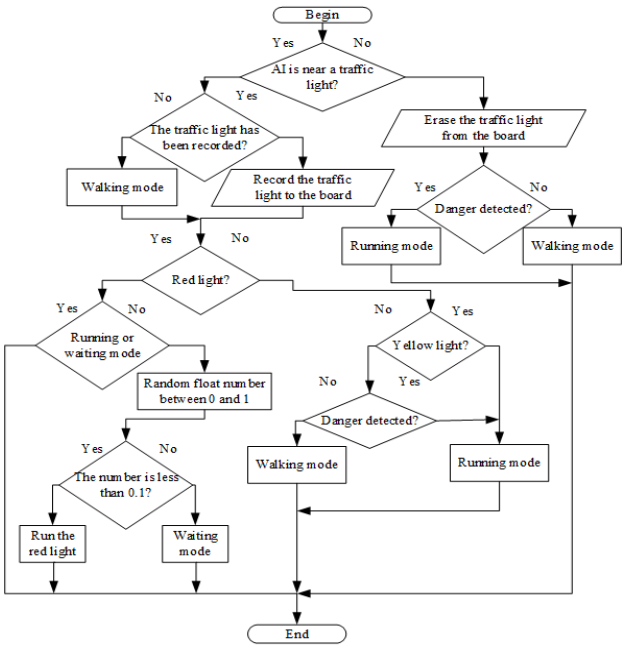

Fig. 3. Service controlling the AI moving mode.

This service is performed throughout the entire AI operation cycle. When the AI is located in a traffic light zone, it checks the light color. The AI will start to run across a road in the last seconds of a green light. Having entered a traffic light zone during an active red light, the AI switches to waiting for the green light mode, but with a 10% probability, it may decide to run a red light. It is necessary to make a novice driver practice his attention and reaction.

If the board indicates that the AI has detected danger, the service will toggle the run mode as soon as possible to take the AI away from the roadway and avoid a hit by a car.

The next node in the behavior tree is the sequence introduced to simulate a large number of pedestrians. In order to save the system resources, the simulator uses a small number of pedestrians for the entire area of the virtual city. The task of this branch is to move the AI closer to the player's car to imitate a densely populated city with only approximately 20 pedestrians in use. The decorator allows the execution of a sequence of tasks only when the car is at far distance from the AI. This decorator interrupts the execution of all branches to the right when the car goes too far from the AI.

The sequence uses an EQS-query that generates a set of points in the direction of the player's movement. Using Tests in the query, the filter cuts off all the points to which it is impossible to make a route, and which are beyond the distance limit. The best option among the remaining set is selected randomly. Figure 4 shows the result of such query.

Fig. 4. A set of points generated by an EQS-query for the player.

The figure shows that the points on the roadway outside the pedestrian zones are cut off and the selection will be made from the points far ahead. When the point is selected, the sequence node to the right of the tree will be executed and activate an instant relocation of the AI.

The next node in the behavior tree is the sequence that performs the logic of selecting the destination point of the route and moves the AI to this point. The selection of the destination point is made by using an EQS-query, similar to the previous one.

This query generates a grid from the AI location covering a certain distance. All points to which it is impossible to make a route and which are closer than a given distance are cut off by means of Tests. The best destination point is the most remote one. Figure 5 shows the result of this query.

The query to find the destination point for the AI is similar to those described before. In this case, the AI generates a set of points in all directions around itself and selects the most suitable point at a distance not less than the specified value.

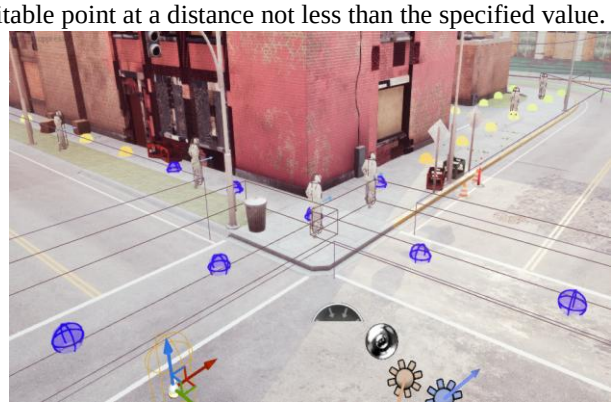

Fig. 5. A set of points generated by an EQS-query for the AI.

This query is followed by the task "Move To" that makes the AI start moving towards the point selected through the EQS-query. As the AI moves to the selected point, the service checks whether it is in a traffic light zone, and the decorator attached to this task will interrupt its execution, in case the pedestrian starts to wait for the green light. The far right branch implements the logic of waiting for the green light.

virtual pedestrian will play a random animation while waiting for the green light and having reached his destination.

VI. CONCLUSION

By means of the tools included in Unreal Engine 4 we have managed to implement a low-resource demanding pedestrian AI system that creates the effect of a densely populated city with a small actual number of pedestrians in use.
The AI reacts to traffic lights, understands when he should go faster and when he should wait. Utilizing the component of sight and hearing, we have managed to implement the danger detection and the actions to avoid being hit by a car. The AI can change the direction of its movement randomly and go in the opposite direction. With a 10% probability the AI can run a red light, which will make a driving school student always be careful while driving a virtual car.

REFERENCES

1. Blueprints - Web resource: https://docs.unrealengine.com/latest/INT/Engine/Blueprints/GettingStarted/index.html
2. Behavior tree - Web resource: https://docs.unrealengine.com/en-US/Engine/AI/BehaviorTrees/QuickStart/14
3. Black Board - Web resource: https://docs.unrealengine.com/en-US/Engine/AI/BehaviorTrees/QuickStart/5
4. AI Perception component - Web resource: https://docs.unrealengine.com/en-us/Engine/Components/AI
5. Selector and sequence - Web resource: https://docs.unrealengine.com/en-US/Engine/AI/BehaviorTrees/QuickStart/10
6. Decorators - Web resource: https://docs.unrealengine.com/en-US/Engine/AI/BehaviorTrees/QuickStart/13
7. Services - Web resource: https://docs.unrealengine.com/en-US/Engine/AI/BehaviorTrees/QuickStart/11
8. Environment Query System - Web resource: https://docs.unrealengine.com/en-US/Engine/AI/EnvironmentQuerySystem/UserGuide

AUTHORS PROFILE

Fares Abu-Abed Ph.D., Associate Professor, Dean of the Faculty of International Academic Cooperation. Associate Professor at the Department of Electronic Computers at Tver State Technical University. Has more than 200 works and patents in the field of IT-technologies and artificial intelligence. aafares@mail.ru

Alexey Khabarov Ph.D., Associate Professor, Dean of the Faculty of Information Technology. Professor at the Department of Electronic Computers at Tver State Technical University. Has many publications and research manuals in the field of IT-technologies and information security. al_xabarov@mail.ru