A Vision Simulation Algorithm for Non-Player Character in Static Scene

Department of Technical computer science

Researchers: M.Sc. Ibrahim Mahmoud
M.Sc. Yasser Jaffal
Advisor: Prof. Dr.-Ing. M. Eng. Dieter Wloka

19, September 2013
Contents

• Introduction and Objective
• Synthetic Vision
• Model and Algorithm
  ○ Vision Module
  ○ Knowledge Module
  ○ Navigation Module
• Experiment
• Future Work Directions
  ○ Dynamic environments.
  ○ Memory and learning.
Character-Based AI

- An AI category that is especially meaningful in games.
- Systems that seek to simulate the behavior of an agent.
- Exercise in creating complete brains.
- Life-like behavior.
Avatars or none player characters (NPCs) are important part for the success of computer games.

Most of the current computer games provide NPCs that are completely controlled by the game logic and react to actions in a pre-defined action-state mechanism.

Players will have access to only limited information about the game.
Game developers can provide the game NPCs with full information about the game, giving them the advantage over the player.

NPCs will not only be able to defeat the player, but also they will look smarter than what they really are.
Example: NPC can know beforehand the best hiding locations in the map of First Person Shooter (FPS) game.

Many game players and even developers, consider it impressive to have each NPC in the game as an accurately simulated creature.
In order to provide such type of NPCs, they should act naturally and perceive their environment using their sensing capabilities, such as **vision**, **hear** and **smell**.

**Visual perception** is our interest, as vision can provide many useful information about the environment in which the NPC operates.
Synthetic Vision

- NPCs “SEE” the world through 2D views of the environment.
- Using these views NPCs are expected to extract useful information and build their knowledge about the game.
- In SV we bypass the classical computer vision problems.
In SV we skip the problems of distance detection, pattern recognition and noisy images.

Using SV:
- **Depth perception**: supplying pixel depth as part of the SV of NPC’ vision.
- **Object recognition**: supplying object function or identity as part of the SV system (Tagging).
- **Motion detection**: coding the motion of object pixels into the SV view-port.
The algorithm consists of three main modules;
- Vision module,
- Knowledge module,
- and Navigation module.
Vision Module

- Our NPC vision algorithm has these constraints:
  - Limited view angle.
  - Can’t “see” behind (through) objects.
  - Can “see” part of the object in his field of view.
  - Can’t “see” very far away objects.

- The visibility is implemented in the vision module in three steps. As follows.
First, calculate the NPC’s cut-out frustum for objects fall within the NPC’s view frustum.
Second, calculate the field of view for the NPC.

- If the angle between the vector from NPC to the object and the NPC forward vector is less than half of the NPC’s view angle, then the object is in the field of view of the NPC.
Finally, ray casting from the NPC to the object in question, to make sure nothing is blocking the view.
Knowledge Module

- This module is used to store information about new objects the NPC “see”.

- Knowledge started to be empty.

- When the NPC “see” an object, it checks if it is already exists in the knowledge, if not, it’s added.

- Information stored includes;
  - Object tag,
  - Object current location.
  - Object correct position.
  - Object priority.
The knowledge module provides some useful methods that help the NPC to achieve his intended goals.

Knowledge is updated as long as the NPC is operating in the environment and it keeps recording events and updates.
Navigation Module

- Navigation module is problem dependent.
- It retrieves whatever information provided in the knowledge module at the time being and use it to select the best navigation task the NPC should make.
- The navigation module guides the NPC to achieve his goals in the environment using his current knowledge to select best action to do.
• We choose to implement our algorithm into a project where the NPC will have the task of loading equipments into a fire truck.

• A simple environment built including two fire trucks (one empty and one full) and some shelf.

• The equipments are randomly scattered over the shelf.
The NPC mission is to collect the fire truck equipments and carry them piece by piece to its correct position on the fire truck, and place it there.

When the application starts, the NPC will have no information about the environment, nor the positions of the game objects.

He is supposed to start navigating and building his knowledge-base from what he can “see” in the environment and record these information.
Our NPC has the following pseudo-code:

- Look for empty vehicle and save its location.
- Look for shelf and save their locations.
- While shelf are not empty, do:
  - Look for objects.
  - Update your knowledge-base.
  - Select object with highest priority.
  - Carry object to its correct place in the vehicle.
  - Go back to shelf.
  - Update knowledge-base.
- Navigate back to the start position.
Exp. Cont.
Future work directions

- Our future work will involve dynamic environments, moving objects.
- Providing our NPC with some sort of memory mechanism. Long term memory and short term memory.
- Providing our NPC with learning algorithm so he can adapt to his environment and show a progress in action selection over time.
Thank you for listening

Questions?!