CMSC 425: Lecture 16
Artificial Intelligence for Games: Basics
Thursday, Apr 4, 2013

Reading: Some of the material from today's lecture comes from the book "Artificial Intelligence for Games" (2nd Edition) by I. Millington and J. Funge.

What is Artificial Intelligence? *Artificial intelligence* (AI) can be defined (circularly) as "the study of computational systems that exhibit intelligence." Unfortunately, it is not easy to define what we mean by "intelligence." In the context of games, an in particular in the design of non-player characters (NPCs) a working definition might be, "It is whatever a person would do." (Where, of course, the word "person" might be replaced by "ogre," "zombie," or "enchanted unicorn," whatever makes sense for the current context.)

At a basic level, game entities have *goals* that they are expected to achieve (e.g., staying out of danger, pursuing the enemy, fighting). This leads to a view of AI as planning strategies to achieve these goals. Computing optimal ways of achieving these goals may involve to *optimization algorithms* at a low level and lead to complex *planning strategies* at a high level. Often, in games, AI is most evident in games when it fails, that is, when other rationally behaving characters behave in an inexplicably unintelligent manner. (For example, it may fail to find a path around an obstacle, when such a path is visually obvious.)

Roles of Game AI: Generally, AI is used in games is to determine complex behaviors that not specified by the player. Examples include:

Nonplayer Opponents: For example, in an FPS, opponents should exhibit realistic attack behavior, which might include a decreased level or aggression or even retreating when suffering damage. This

Nonplayer Teammates: For example, given a squadron of soldiers, the group should move in a coordinated supportive manner. Such support NPCs are sometimes employed in multiplayer online games to assist inexperienced players. In some contexts, this might be scripted by the game designer. In others, the motion would be computed through the use of AI.

Support and Autonomous Characters: This includes generating realistic crowd behavior, where the characters may need to interact in a realistic manner when coming into contact with the player's character.

Commentary/Instruction: Again, this is typically scripted, but an example requiring AI might involve determining whether the player is stuck and in need of a hint on how to proceed.

Camera Control: Typically camera control is simply automatic, but in a complex environment it may require intelligence to compute a good viewpoint, from which it is possible to identify the important elements of the environment and is not obscured by obstacles.

The key element in all of these examples is the feature of *complexity*. Examples of things that are *not* AI include:

Determined by physical laws: Examples include the flight of a tennis ball or the reaction of a car that hits an obstacle.

Purely random: An example would include which block falls next in Tetris.

Direct response to game rules/user inputs: This includes events for which the response is predetermined by the game designer. This includes typical camera control, scripted animations, events that are triggered by the user's inputs, and events that are scheduled to occur at a particular time or after a particular time delay.

One notable gray area is where AI ends and animation begins. For example, a soccer player dribbling the ball must make decisions as to how to avoid opponents, which in turn affects the direction and speed with which he runs, which in turn affects joint angles. Typically, AI systems control the high-level decisions and the animation controls the lower level decisions:

Should I run with the ball or pass it? This is definitely an AI decision (unless it has been scripted)

If I run, what path should I take? This is getting into the gray area. If we wish to evaluate the likelihood of success of various options, based on hypotheses of how the player and other NPCs might respond, we are definitely in the realm of AI. If we simply wish to compute a shortest obstacle-avoiding path (say using Dijkstra's algorithm), this is in the realm of *algorithmics*[1]

How to move my legs to travel along this path? Now we are definitely outside of the realm of AI, and into the realm of animation.

Properties of a Good AI System: The following is a list of generally good properties of a game AI system.

Goal driven: The AI system should behave in a manner that is consistent with the (implicit) high-level goals of the entities involved.

Responsive: The AI system should respond rapidly to relevant changes in the state of the world. For example, if a path is blocked, the NPC should respond quickly by computing a new path.

Smart, but not omniscient: The AI system should behave as if it knows a good deal about the world (inanimate objects, other NPCs, and even the player) and select its behaviors accordingly. Of course, an NPC cannot act based on information that it could not reasonably have knowledge of.

Consistent: An NPC should behave in a consistent manner, to generate the impression that it embodies a believable character.

Efficient and Practical: Computational resources are limited, and the time needed to develop, program, and test the AI system must be considered within the economic constraints of the game.

Unfortunately, many of these goals conflict with each other, and many of the problems in game AI result from developers making compromises in quality for the sake of simplicity or efficiency.

The AI Loop: The fundamental view of a game from the perspective of AI involves continuously evaluating the current state of the (perceptible) world and determining what should the agent do in the very near future (that is, the next frame to be drawn). This generally may involve a number of layers of sensing and decision making. These are illustrated in Fig. 1.

Perception: The entity senses the elements of the game state that is within its scope of perception.

[1]Some might claim that this is AI, since algorithmics is just an offshoot of AI. If you say this in earshot of an algorithms researcher, be prepared to get punched in the face.

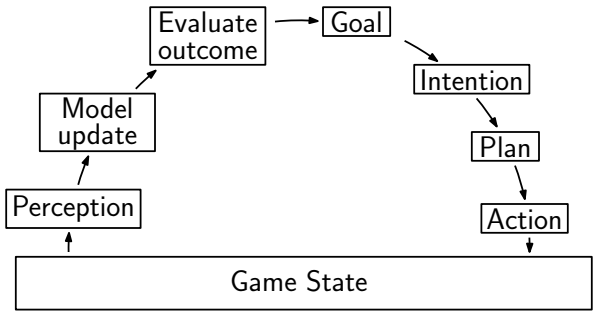

Fig. 1: The game AI loop.

Model update and outcome: The perception of the state results in updates to the entities internal state model. The outcome of these state changes may be very simple ("I am injured and need to retreat") but could conceivably be quite sophisticated in the context of a game with complex narrative structure ("An ally has acted against my interests. Is he a spy?")

Goals and intentions: Based on a character's understanding of the world state, what are its motivations, and how are these motivations weighed to arrive at a set of goals? These goals need to be mapped these into intentions that are to be manifest through the character's future actions. ("That sniper is killing too many of us. How can we reduce our visibility? Hiding? Shoot out the lights? Smoke grenades?")

Plan and action: Given these intentions, the character then needs to develop a plan of actions in order to achieve the desired results. Such a plan consists of a sequence of tasks. Once a plan has been developed, the character needs to act in order to perform these tasks.

Agents: NPCs are often modeled in games through the use of an AI construct called an agent. An (autonomous intelligent) *agent* is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

At a high conceptual level, an agent is characterized by three basic components, which operate iteratively over time:

Sensing: Perceive features of the environment, and in particular, changes to the environment that are relevant to the agent's goals.

Thinking: Decide what action to take to achieve its goals, given the current situation and its knowledge. Of course, this is where all the complexity lies. Thinking may be simple reaction ("Danger. Flee!") or may involve complex responses based on past experiences and learning. If the objectives are complex, apply planning strategies to break them into well-defined actions.

Acting: Carry out these actions.

One can develop a taxonomy of agents, based on their sophistication.

Simple reflex agent: acts solely based of the current perception without regard to previous experiences. These are usually implemented by simple *rule-based systems*. "If X occurs then do Y." These are often encoded as a table, where input events are mapped to integer indices and the ith table entry is an encoding of the action to take on stimulus i.

Model-based agent: extends the simple reflex agent by storing its own internal state ("I'm healthy, hungry, injured, etc."), the state of the perceived world ("I hear a threat approaching from the east"), and some model of how the world works.

Such an agent chooses an action in the same way as the reflex agent, but the action will generally depend on the model's current state. These agents are often implemented by a *finite-state machine*. The machine's state corresponds to the agent's state, and current perception triggers an action and a transition to a possibly different state (For example, "If I am wandering and healthy (state) and see the player's avatar (current perception), I will start pursuing it (action).")

Goal-based agent: further extends on the capabilities of the model-based agents, by using *goal information*. Goal information describes situations that are desirable. This allows the agent a way to select among multiple possible intentions, selecting the one that will reach a goal state. Search and planning algorithms may be invoked to map goals into low-level actions.

Utility-based agent: further extends the goal-based agent by defining a measure of how desirable a particular state is. This measure is expressed through the use of a *utility function*, which measures how happy the agent would be with this state. The agent then chooses the action that maximizes the expected utility of the action.

Learning agent: takes all of this a step further by evaluating the results of past action, and uses this information to make (hopefully) better choices in the future. It has two important components, a *learning element*, which is responsible for making improvements by critically evaluating the benefit of the outcomes of prior actions, and *performance element*, which is responsible for selecting external actions. Note that future actions may not be chosen solely on the basis of expected utility, but there may also be exploration of unknown states to determine their utility.

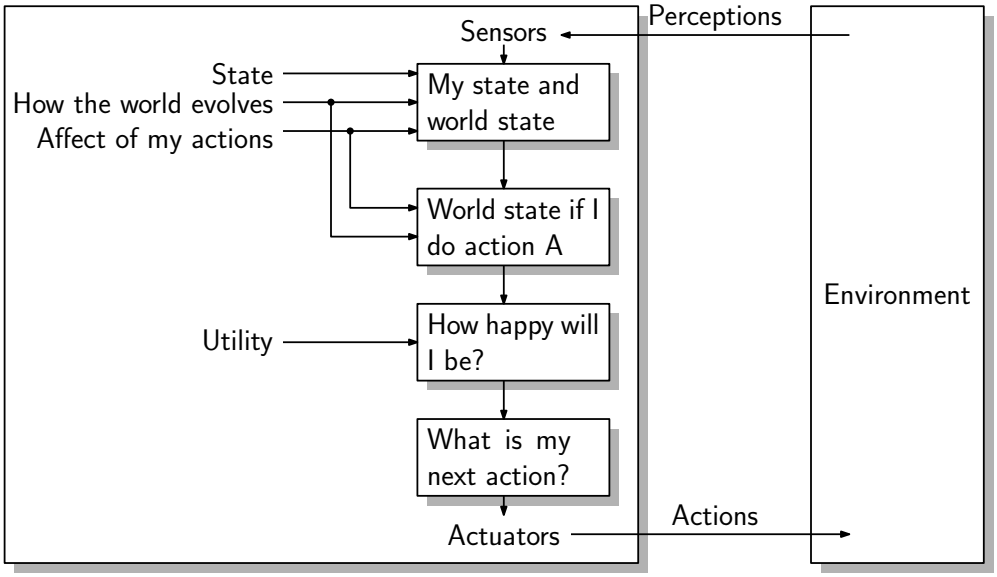

Fig. 2: A utility-based agents.

Looking ahead: In future lectures, we will explore two important aspects of AI in computer games. The first is planning motion to achieve various goals (e.g., pursue the enemy by the shortest path) subject to various constraints (e.g., avoid obstacles). We will also discuss techniques for making decisions.