Artificial Agents

Canberra Institute of Technology

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Lecture Contents

- What is an Agent
- Sense, Think, Act
- Behaviours
- Combining Behaviours
- Basic Implementation

What is an Agent?

An Agent is a game object that has some sort of intelligent behaviour and or decision making process.

There are 3 common steps which are repeated during an agents “Update” process.

- Sense
- Think
- Act

Sense

The sense step is where the agent analyses the world and checks for changes to the environment.

This could include (but not limited to)

- Detecting potential future collisions
- Detecting changes in behaviour of other agents
- Are other friendly agents in-danger?
- Calculating values such as “Hungriness”, “Scaredness” and other such values that could be plugged into a “fuzzy logic” system.
- Where any sounds played near by that the agent may want to respond to.

This step just collects the information required to make a decision during the “Think” step.

Think

Once the agent has gathered information, it can now be evaluated and a decision can be made.

This step can be incredibly simple or insanely complex.

This could consist of handcrafted if then rules to determine which behaviour should be used through to a more complex decision making process utilising various AI systems such as: Fuzzy Logic, Decision/Behaviour trees, Machine learning algorithms etc.

Act

Until now, the end user will not have been able to witness the slew of decisions that have been made.

The Act step is used to preform the appropriate behaviour that had been calculated during the “think” step.

For example: if the think step decided that a “Seek” behaviour should be used, than the “Act” step will preform the seek operation.

Behaviours

Agents should have the ability to change between various different behaviours. We can think of a behaviour as logic used to preform a single task.

For example:

- Seek, Flee, Wander, Follow Path, Avoid, Patrol.
- ShootWhenInRange, DropBomb, ClimbLadder, CollectItems<type>

Combining Behaviours

It should be possible to combine behaviours together which will assist with making an agent appear more intelligent.

For example:

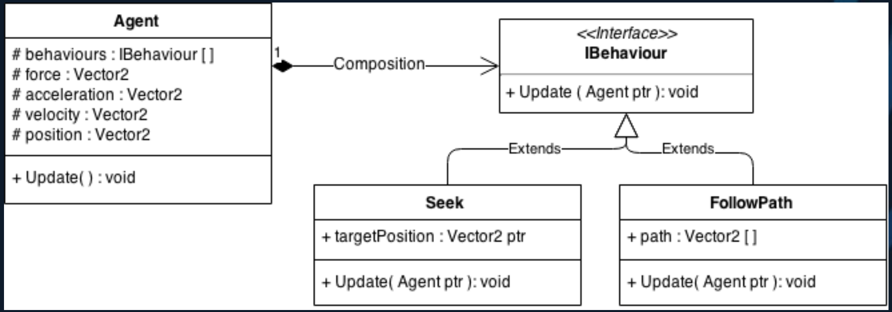
- Seek – seek toward player
- Flee – flee from enemy objects (except player)
- ShootWhenInRange – if the player is in range, shoot at it.

Basic Implementation

Its common practice to wrap an individual behaviour into a single class.

Define an abstract Behaviour class for all other behaviours to extend

An Agent can have a collection of behaviours. Each time the agent preforms the “Act” it should run the behaviour’s Update method, passing “this” in as the parameter



Basic Implementation (C++)

```cpp
class Agent
{
public:

    void Update()
    {
        // TODO: Sensing Calculations
        // TODO: Thinking Calculations
        //       Add or remove behaviours from the m_behaviours list

        // Acting:
        for(auto iter = m_behaviours.begin(); iter != m_behaviours.end(); iter++ )
            (*iter)->Update(this);

        //TODO: Physics stuff  with force, acceleration, velocity etc...
    }

protected:

    std::list< IBehaviour * > m_behaviours;

    Vector2 m_force;
    Vector2 m_acceleration;
    Vector2 m_velocity;
    Vector2 m_position;
};
```

```cpp
class IBehaviour
{
    virtual void Update( Agent *pAgent ) = 0;
};
```

```cpp
class Seek : public IBehaviour
{
    Vector2 *m_targetPos;

    virtual void Update( Agent *pAgent )
    {
        // TODO: apply seek force to pAgent.
    }
};
```

Canberra Institute of Technology

aie

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Other uses

With this particular implementation, we can create behaviours that aren’t necessarily classified as AI.

For example, we could create a “KeyboardController” behaviour, which collects input from the user and moves the Agent based on keyboard input.

Other uses

We could create a behaviour with a single set of properties that is shared amongst multiple Agents.

For example, you may have a collection of agents who are following the same path. They could all share the same “Follow Path” behaviour instance.

Side Node: you will want to look at using smart pointers when sharing behaviours, as no single agent will have ownership over the behaviour.

Questions?

References

- Steve Rabin, 2009, *Introduction to Game Development (second edition) [chapter 5]*