**INSTALLATION FOR PANDAI v0.2**

**Following are the preliminary steps for using the Pandai library:**

Download PandaiV0.2.zip from HERE

Unzip the file and you should find libpandaai.dll.

Copy libpandaai.dll and BuildDLL.bat to ${Your Panda3D Installation}\bin\

Eg: C:\Panda3D-1.6.2\bin\ Run BuildDLL.bat

If there are no errors you should be ready to use Pandai version 0.2 in Python.

**Using Steering Behaviors in Python**

(Note the structure has been changed since the last release, but is still pretty easy)

In your python code,

You will need to import the library:

from libpandaai import \*

Then you will have to create an object of the AIWorld class:

aiWorld = AIWorld(render)

Then you will have to create an object of the AICharacter class:

aiChar = AICharacter(int Character\_id, NodePath model\_nodepath, double mass, double movement\_force, double maximum\_force)

These parameters will determine how your character moves in the world.

Then you will have to attach this character to your AIWorld:

aiWorld.addAiChar(aiChar) // Note: aiWorld and aiChar used from above.

Now you will need to get a handle to the steering behaviors of the AICharacter:

aiBehaviors = aiChar.getAiBehaviors()

With this handle all the 5 steering behaviors can be called as such:

**NOTE:**

All the behaviors take a priority value which will be explained in detail after their descriptions.

Also, the movement forces are defined in the character above and we will give you control over helper functions to modify them if required.

**SEEK:**

aiBehaviors.seek(NodePath target, int priority)

aiBehaviors.seek(Vec3 position, int priority)

'Seek' takes the target or a position to seek. Seek's direction is calculated only during the first instance of the call and so is more efficient than pursuit, if all you want is an object to go to a point.

**FLEE:**

aiBehaviors.flee(NodePath target, double panic\_distance, double relax\_distance, int priority)

aiBehaviors.flee(Vec3 position, double panic\_distance, double relax\_distance, int priority)

'Flee' takes in a target or a position to be fled away from; this position should be static. (For moving objects use Evade).

Panic Distance is the radius of detection.

Relax Distance is the distance away from the panic distance after which the object should stop fleeing once flee has been initiated.

**PURSUE:**

aiBehaviors.pursue(NodePath target, int priority)

'Pursue' takes the target to be pursued. The direction is continuously calculated here and this behavior would represent a continuous seeker or follower.

**EVADE:**

aiBehaviors.evade(NodePath target, double panic\_distance, double relax\_distance, int priority)

'Evade' takes in a target to be fled away from, which can be moving.

Panic Distance is the radius of detection.

Relax Distance is the distance away from the panic distance where the object should stop fleeing once flee has been initiated.

**ARRIVAL:**

aiBehaviors.arrival(NodePath target, double distance, int priority)

'Arrival' is a behavior which is usually used with seek or pursuit so that the object slows down on approach to its target.

**Priority**

Priority is a new feature which we have added to our behavior system. This feature allows a user to determine the resultant force when there is a conflict between two or more behaviors.

For example: Incase 'A' is pursuing 'T', 'B' is pursuing 'T' and 'A' is also evading 'B'.

This will result in a conflict when A is close to B as it has to decide whether to seek or evade. This conflict can be solved by determining the priority.

Priority is a value between 0 and 1 which you specify in your behavior functions.

For example: If pursuit had a priority of 0.1 and evade had a priority of 1.0, then on conflict the AICharacter will definitely evade with more force than pursuit.

**AIWorld Update**

Finally, you will need to run the AIWorld update which is a global AI update for every character in the world, in a continuous task.

aiWorld.update()

Note how now there is only one function for our entire AI that you need to use a task for and so it is way more convenient.

Doing the above steps correctly will ensure that Pandai will work properly in your world.

**Helper functions**

This is a section we made to ease the job of scripters using our AIs:

For the AICharacter class:

double getMass();

void setMass(double m);

LVecBase3f getVelocity();

double getMaxForce();

void setMaxForce(double max\_force);

NodePath getNodePath();

void setNodePath(NodePath np);

**For the AIBehaviors class:**

The status functions which can be called in a task continuously to find whether a function is going on (false) or is done (true).

bool seekStatus();

bool fleeStatus();

bool pursueStatus();

bool evadeStatus();

bool arrivalStatus();

**To remove any AI after their call has been instantiated:**

void remove(int ai\_values);

0- removes all the Ai's

1- removes seek

2- removes flee

3- removes arrival

4- removes pursue

5- removes evade