FlatworldII V1.0 Simulation Environment

API Description

Thomas P. Caudell
University of New Mexico
Neural Networks Class Version
November 2009 Draft

1- General Description

Flatworld is a simulated two-dimension world for the training and testing of one or more simulated agents. Time dependent actions in the world are based on a discrete time step model. The world is populated with two-dimensional geometrical objects, some of which are eatable by the agents, some of which act as various kinds of barriers. The agent and objects have physical properties such as shape, color, food value or potential energy, position and velocity, and mass. An agent has a polygonal body and a circular head.

Agents have sensors, both internal and external, and actuators. Sensors include eyes, ears, skin contact sensors, muscle actuator states, and internal energy charge. Actuators include muscles for movement within the world (forward-backwards, left-right, change in heading), muscles for changing head direction with respect to the body, sound emission, object pick up/put down, and eating. The agent has a basil metabolism that reduces an agent's energy charge each time step, as well as a muscle metabolism that reduces the energy charge as a function of movement step size.

Some sensors, such as light have a limited distance range, whereas others, such as sound have no range limits. Vision and sound signal strength diminishes with distance, similar to the real world. The light (vision) sensor (eye) is composed of individual light receptors complexes (eyelets) that sense red, green, and blue optical reflectivity's of objects or other agents along a ray direction. The reflectance value is computed at the intersection point of the eyelet ray and a polygonal segment of an object or agent. For a given eye, eyelets are located on the surface of the circular head defined by their angular position relative to the head with the default direction pointing radially outwards from the center of the head. Each eyelet my have its own pointing direction relative to its radial pointing direction. Ears are also located on the surface of the head. The sound (hearing) sensor (ear) is composed of individual sound receptors complexes (earlets) that can detect the amplitude and (in the future) the phase of sound at a given frequency. Sounds form all object and agents are combined at an ear.

The contact sensor (skin) is composed of a set of linear polygonal receptors that are located coincident with the exterior of the agent's body that measure properties of contact with world objects/agents. Currently the only property measured is the binary detection of intersection with other polygonal segments of other object/agents.

2- Coordinate Systems

Flatworld is a two spatial dimensional environment and therefore relies of coordinate systems. Figure 1 gives an overview of the three nested coordinate system used in this system. The first is the world system consisting of an x-y plane where objects and agents are located and are confined to. The second is the agent body system in which skin cells are located and in which movement occurs. Note that movement steps are relative to the body coordinate system. The final system is the head coordinate system illustrated in Figure 2. Eyes and ears are located along the surface of the head defined by angles relative to the "nose" direction.

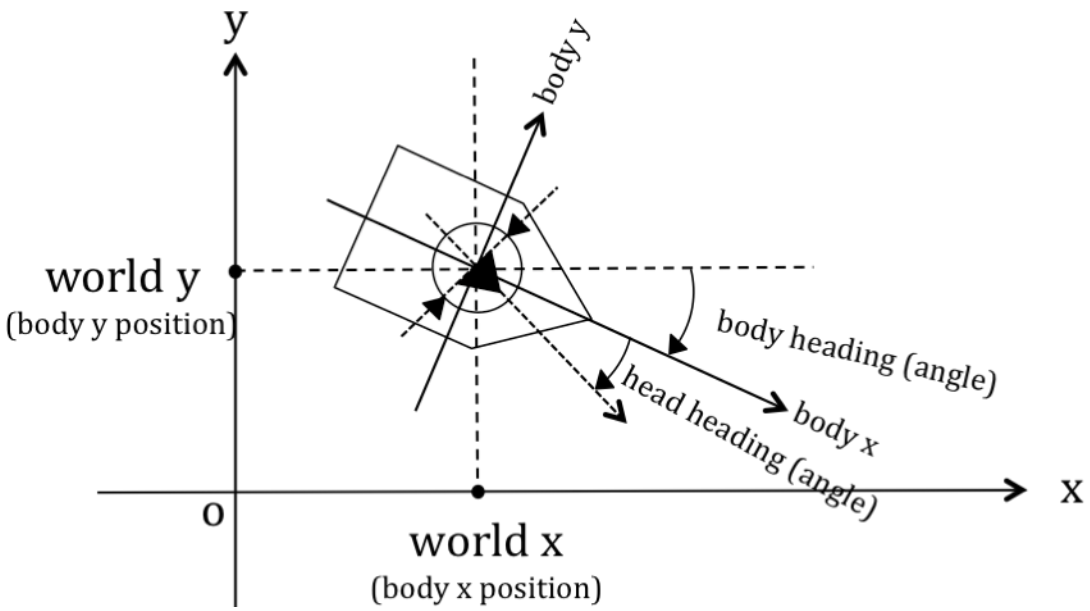
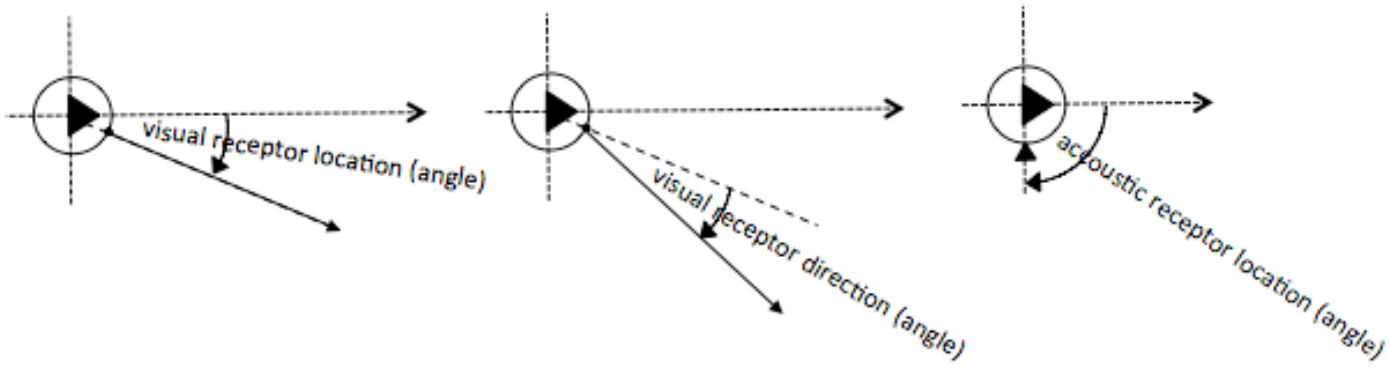


Figure 1. General coordinate systems. The three major coordinate systems are 1) the world system, 2) the agent's body system, and 3) the head system. The five-sided polygon is the body of the agent. The circle is the agent's head, and the black triangle within the head points in the head's "nose" direction.



(a) (b) (c)

Figure 2. Head related coordinates. (a) the location of eye receptors in degrees, (b) the pointing direction of an eye receptor in degrees relative to the radial pointing direction, (c) the location of an ear in degrees.

3. Shape Files

3.1 Use of Shapes in the Simulation

Shapes are used for specifying properties of the agents and objects. For example, the geometric shape of an agent or an object is a polygon specified by a geometric shape file. Geometric shape files are flat text files consisting of a set of parameters and a list of 2D vertices of the shape plus heir RGB colors. Each vertex may have a different color. Sound shape files are flat text files that specify the shape of the frequency spectrum of the agent or objects emitted sound. A

sound shape file consists of a set of parameters and a list of sound amplitudes at different frequencies. (Note to self: add frequencies to structure and files as well)

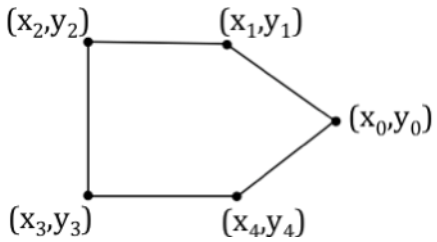


Figure 3. An example of a geometric shape.

3.2 Shape file formats

These files are flat text files usually with a *.dat* extension.

Geometric shape files:

nvertices ncolorbands z-height scale
x1 y1 color1 color2 color3
…..
xn yn color1 color2 color3

Sound shape files:

nfrequencies nbands
freq1 amp11 amp12 … amp-1,nbands
…
freq-nfreqs ampn1 ampn2 … amp-n,nbands

3.3 Reading Shape files

```
GEOMETRIC_SHAPE_TYPE *read_geometric_shape_file( char *filename )
ACOUSTIC_SHAPE_TYPE *read_acoustic_shape_file( char *filename )
```

4- Agent Creation

4.1 Agent Description

In Flatworld, and agent is first created, then attributes are added through a series of function calls. After an agent is fully attributed, it is added to the world.

4.2 Agents Creation

```
AGENT_TYPE *make_agent( int index, float x, float y, float body_angle, float head_radius, float
mass, float metabolic_burn_rate )
void free_object( OBJECT_TYPE *o )
```

4.3 Adding Agent Attributes

```
void add_physical_shape_to_agent( AGENT_TYPE *a, GEOMETRIC_SHAPE_TYPE *shape )
void add_sound_shape_to_agent( AGENT_TYPE *a, ACOUSTIC_SHAPE_TYPE *shape )
void add_visual_sensor_to_agent( AGENT_TYPE *agent, int nreceptors, int nbands, float
initial_values, float *receptor_locations, *receptor_directions )
```

```
void add_acoustic_sensor_to_agent( AGENT_TYPE *agent, int nreceptors, int nbands, float
initial_values, float receptor_location )
void add_cargo_manifest_type_to_agent( AGENT_TYPE *a, int maxnitems )
void add_soma_sensor_to_agent( AGENT_TYPE *agent, int nbands, float initial_values,
GEOMETRIC_SHAPE_TYPE *receptor_locations )
void add_actuators_to_agent( AGENT_TYPE *a, float dfb, float drl, float dth, float dh )
void add_proprio_sensor_to_agent( AGENT_TYPE *agent, int nreceptors, int nbands, float
initial_values, float *receptor_locations )
```

4.4 Adding Agent to World

```
void add_agent_to_world( WORLD_TYPE *world, AGENT_TYPE *agent )
```

5. Object Creation

5.1 Object Description

In Flatworld, and object is first created, then attributes are added through a series of function calls. After an object is fully attributed, it is added to the world.

5.2 Object Creation

```
OBJECT_TYPE *make_object( int index, int type, float x, float y, float mass, float food_value )
```

5.3 Adding Object Attributes

```
void add_physical_shape_to_object( OBJECT_TYPE *object, GEOMETRIC_SHAPE_TYPE *shape )
void add_sound_shape_to_object( OBJECT_TYPE *object, ACOUSTIC_SHAPE_TYPE *shape )
void add_behavior_to_object( OBJECT_TYPE *object, void *behavior_func )
```

5.4 Adding Object to World

```
void add_object_to_world( WORLD_TYPE *world, OBJECT_TYPE *object )
void delete_object_from_world ( WORLD_TYPE *w, OBJECT_TYPE *o )
```

6. Reading Agent Sensors

6.1 Vision

`void read_visual_sensor( WORLD_TYPE *w, AGENT_TYPE *a)`
- reads all visual sensors (or eyes).

`float **extract_visual_receptor_values_pointer( AGENT_TYPE *a, int eye_index)`
- extracts 2D float array of sensed color values. Eye_index is the index of the eye.

`float visual_receptor_position( VISUAL_SENSOR_TYPE *eye, int recindex )`

- this function returns the pointing direction of an individual visual receptor with index `recindex`, in head coordinates

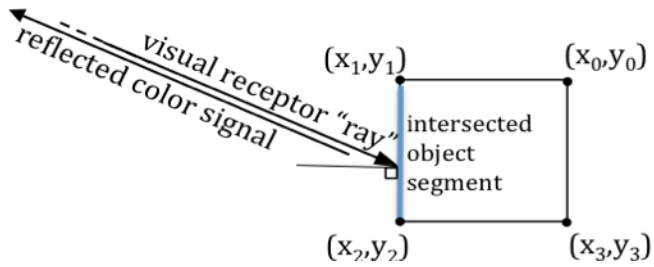


Figure 4. Details of how an individual eyelet receptor calculates its reflected values. The square box represents a simple world object and the ray is coming from one eyelet in an eye on an agent.

6.2 Hearing

```
void read_acoustic_sensor( WORLD_TYPE *w, AGENT_TYPE *a)
```
- reads all sonic sensors (or ears).

```
float **extract_sound_receptor_values_pointer( AGENT_TYPE *a, int ear_index)
```
- extracts 2D float array of sensed sound intensity values. Ear_index is the index of the ear.

6.3 Skin

```
int read_soma_sensor( WORLD_TYPE *w, AGENT_TYPE *a)
```
- reads all skin cells.

```
float **extract_soma_receptor_values_pointer( AGENT_TYPE *a )
```
- extracts 2D float array of sensed skin cell values values.

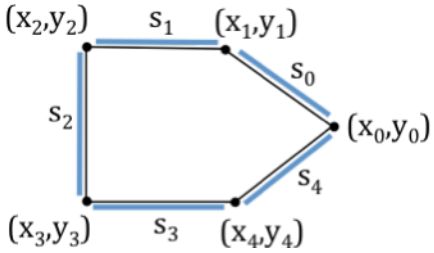


Figure 5. Details of locations of skin receptor cells around the perimeter of the agent's body. In this example, the somatic sensors are coincident with the agent's body shape file.

6.4 Body Attitude

```
void read_actuators_agent( AGENT_TYPE *a, float *dfb, float *drl, float *dth, float *dh )
```
- dfb: delta front-back step (+/-).
- drl: delta right-left step (+/-).
- dth: delta in body heading (+/-).
- dh: delta in head direction (+/-).

```
void read_agent_body_position( AGENT_TYPE *a, float *x, float *y, float *h )
```
- x: body center coordinate (+/-).
- y: body center coordinte (+/-).
- h: body heading (+/-).

6.5 Internal sensors

```
float read_agent_metabolic_charge( AGENT_TYPE *a )
```

7- Causing Agent Actions

To move any part of the agent, first the deltas (velocities) must be set, then the movement command is called.

```
void set_actuators_agent( AGENT_TYPE *a, float dfb, float drl, float dth, float dh )
```
- dfb: delta front-back step (+/-).
- drl: delta right-left step (+/-).
- dth: delta in body heading (+/-).
- dh: delta in head direction (+/-).

```
void move_head_agent( AGENT_TYPE *a )
```
- increments the agent's head angle by dh in body coordinates. This parameter is set in the set_actuators_agent().

```
void move_body_agent( AGENT_TYPE *a )
```
- increments the agent's body center by dfb for the forward/backwards direction, by drl for the right/left direction, and dth for the body heading. These parameters are set in the set_actuators_agent().

```
float agent_eat_object( WORLD_TYPE *w, AGENT_TYPE *a, OBJECT_TYPE *o )
```
- eats all objects in collision with the agent. These objects will disappear from the world. Returns the change in metabolic charge after eating.

```
float agent_eat_object_with_flag( WORLD_TYPE *w, AGENT_TYPE *a, OBJECT_TYPE *o, int flag)
```
- eats all objects in collision if called with flag=1, does nothing otherwise. Returns the change in metabolic charge after eating.

```
void reset_agent_charge( AGENT_TYPE *a )
```
- resets the metabolic charge of this agent to unity (1.0)

```
void set_agent_body_position( AGENT_TYPE *a, float x, float y, float h )
```
- sets the position and orientation of the agent in the world. X and y are world coordinates, and h is the body heading.

8- World restoration

The following function will restore the world back to its original state with all objects visible to the agent:

```
void restore_objects_to_world( WORLD_TYPE *w)
```

9- The Controller.c file

All agent initialization and control can be done from within the Controller.c file and no other file should be modified.

It contains two functions:

```
void init(void)
```
- this function initializes the graphics, the agent and the world.

```
void agents_controller( WORLD_TYPE *w )
```

- this function controls the agent. Note comments that point out what must be replaced with all neural controllers.