SOFTWARE TECHNICAL REFERENCE MANUAL

Contents

REVISION LOG

Sl.No.	Date	Version	The change	Section(s) Impacted
1	11/24/2013	1.0	Baseline created	Nil
2	12/8/2013	2.0	Project Part-3	all

ABOUT

This technical reference is intended to enable the users to install the code, debug it (there will be some in spite of all efforts to get rid of them), expand or modify it .It also has the information needed to install the software Unity™ (user's guide) and run it, and specific and detailed information needed to maintain the code with an intention to facilitate users to access and port the code.

This manual assumes that the user is somewhat familiar with Unity™. If not, it is highly encouraged that one acquaints themselves with it and take a look at its user manual. The installation procedure is as elaborated below.

Instructions for installing and getting started with Unity3D

1. Download Unity3D for Windows or OS X from here : Download
2. Install the application to your computer
    a. Select the free license when prompted to activate Unity™.
3. Follow the instructions to create and login with a Unity™ account.
4. You may want to experiment with an example project to familiarize yourself with Unity™ interface and its functionality.
    a. The example project should be installed in your home directory or Documents directory under "Unity Projects/4-0_AngryBots"
5. Download and extract the code for UAV programming behavior from here :
6. Launch Unity™, select "open project", and select the folder you extracted.
7. For more on Unity™ follow : Help

INTRODUCTION

Like any Unity™ project, the base UAV behavior code consists of assets .The highest level asset in this project is a scene .The other assets include models, textures and scripts. There is one scene defined in this project: “airrobot_block”.It consists of several objects:

- 4 walls and a floor
- “people” (represented by cylinders for the sake of simplicity)
- 3 UAVs (AirRobot)
- 1 “BlockLine” which the UAV(s) attempts to defend.
- 6 cameras (which can be switched
- The people, their attached cameras, and the UAVs all have associated scripts controlling their behavior. Each script is a JavaScript (JS) and the only requirement to ensure its compatibility with Unity™ is the existence of two functions:
    - Start (), which is called when a scene is initialized and
    - Update (), which is called for each new frame Unity™ draws.

The scripts are:

- TheCamera(),to detect people
- Hokuyo(), to detect the position of the UAV.
- PerceptualSchema ()
    - Inputs
        - TheCamera() & Hokuyo()
    - Output
        - Position of the robot and people
- MotorSchema ()
    - Input
        - Perceptual schema; Position of the robot and people
    - Output
        - Behavior/state of the UAV(s)
- LowLevelController ()
    - Input
        - MotorSchema();Behavior/state of the UAV(s)
    - Output
        - Commands to the actuators based on the behavior/state

INSTALLATION OF THE CODE

- To install the code, open the scene (“airrobot_block” in this case) using Unity™.

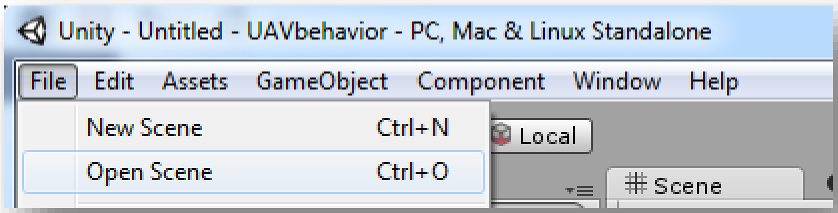


Figure 1

- This opens up the scene as below :

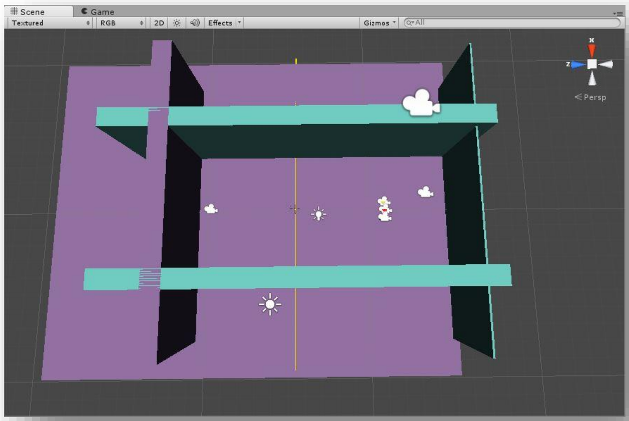


Figure 2

- While the game-objects get listed in the hierarchy panel, the associated scripts appear in the project panel.

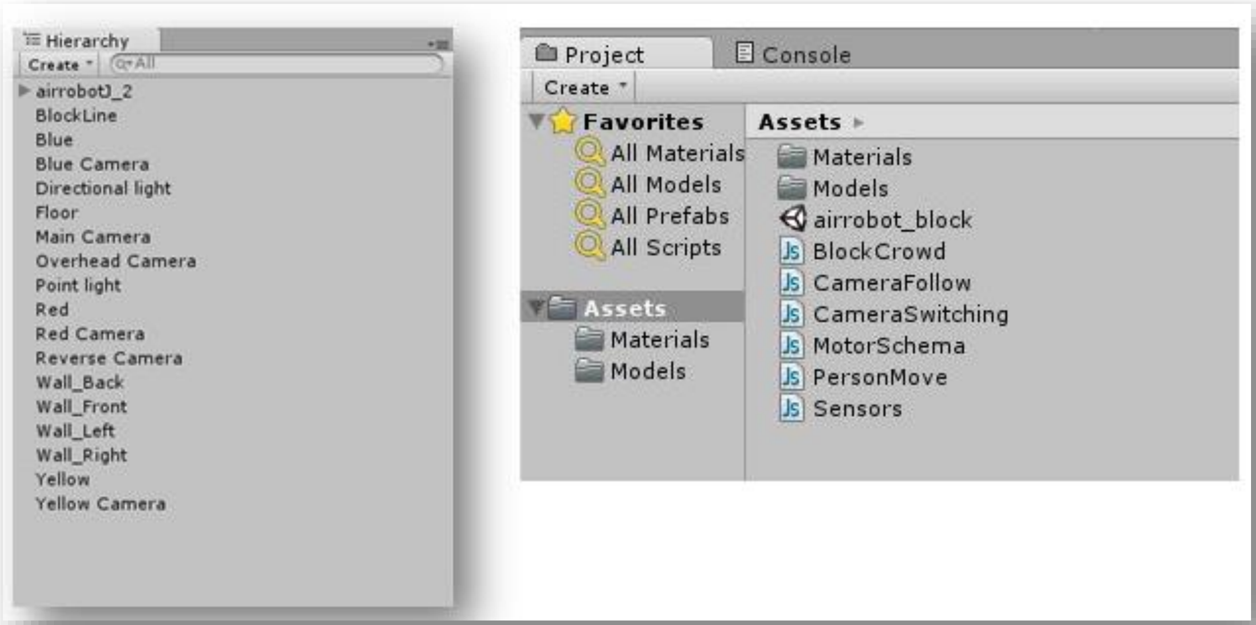
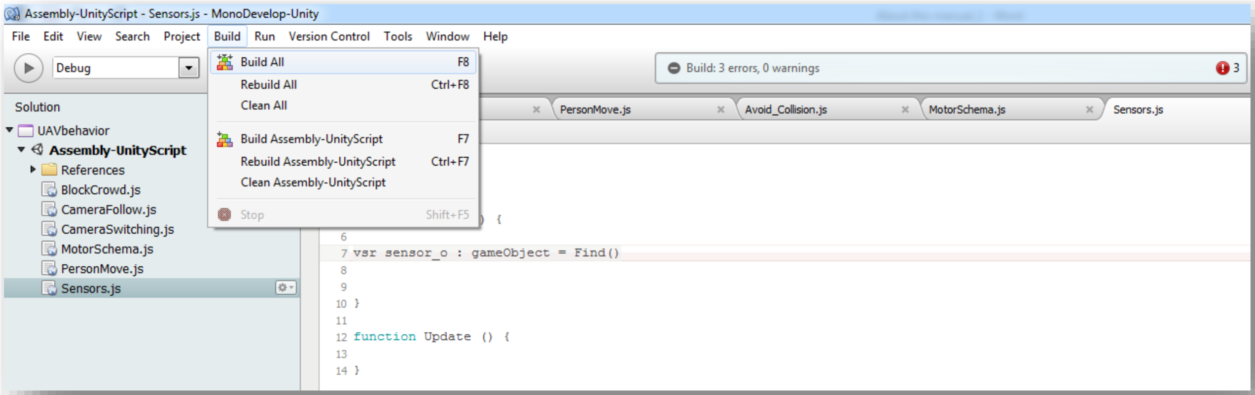


Figure 3

- Each JavaScript can be edited using the editor.



```
) {

vsr sensor_o : gameObject = Find()


}

function Update () {

}
```

Figure 4

- On loading, press on the play button to run.

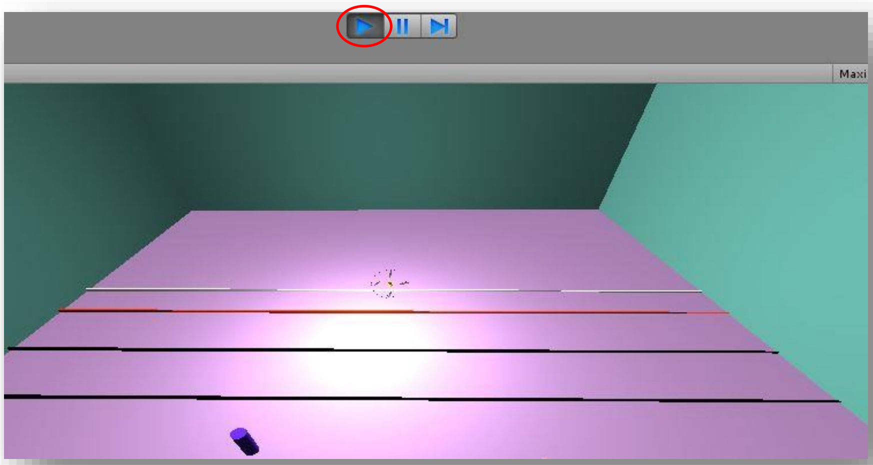
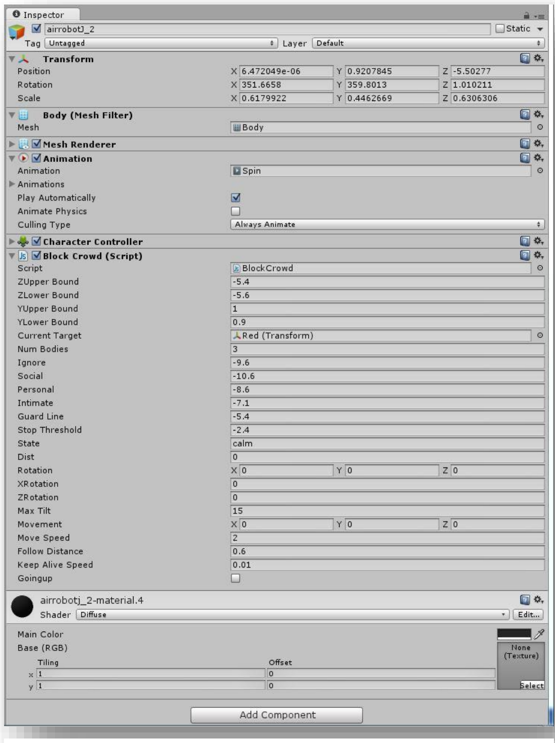


Figure 5

- While the functions and the associated variables in each script get listed in the inspector panel, their value changing dynamically can also be observed on running the scripts.



Inspector

airrobotJ_2 ☐ Static

Tag: Untagged | Layer: Default

Transform

Position	X 6.472049e-06	Y 0.9207845	Z -5.50277
Rotation	X 351.6658	Y 359.8013	Z 1.010211
Scale	X 0.6179922	Y 0.4462669	Z 0.6306306

Body (Mesh Filter)

Mesh	Body

Mesh Renderer

Animation

Animation	Spin
Animations	
Play Automatically	✓
Animate Physics	☐
Culling Type	Always Animate

Character Controller

Block Crowd (Script)

Script	BlockCrowd
ZUpper Bound	-5.4
ZLower Bound	-5.6
YUpper Bound	1
YLower Bound	0.9
Current Target	Red (Transform)
Num Bodies	3
Ignore	-9.6
Social	-10.6
Personal	-8.6
Intimate	-7.1
Guard Line	-5.4
Stop Threshold	-2.4
State	calm
Dist	0
Rotation	X 0 Y 0 Z 0
XRotation	0
ZRotation	0
Max Tilt	15
Movement	X 0 Y 0 Z 0
Move Speed	2
Follow Distance	0.6
Keep Alive Speed	0.01
Goingup	☐

airrobotj_2-material.4

Shader: Diffuse | Edit...

Main Color

Base (RGB)

	Tiling	Offset
x	1	0
y	1	0

None (Texture) Select

Add Component

Figure 6

STRUCTURE OF THE CODE

In an attempt to describe the structure of the code, the design and the software implementation are described in the following sub-sections.

DESIGN

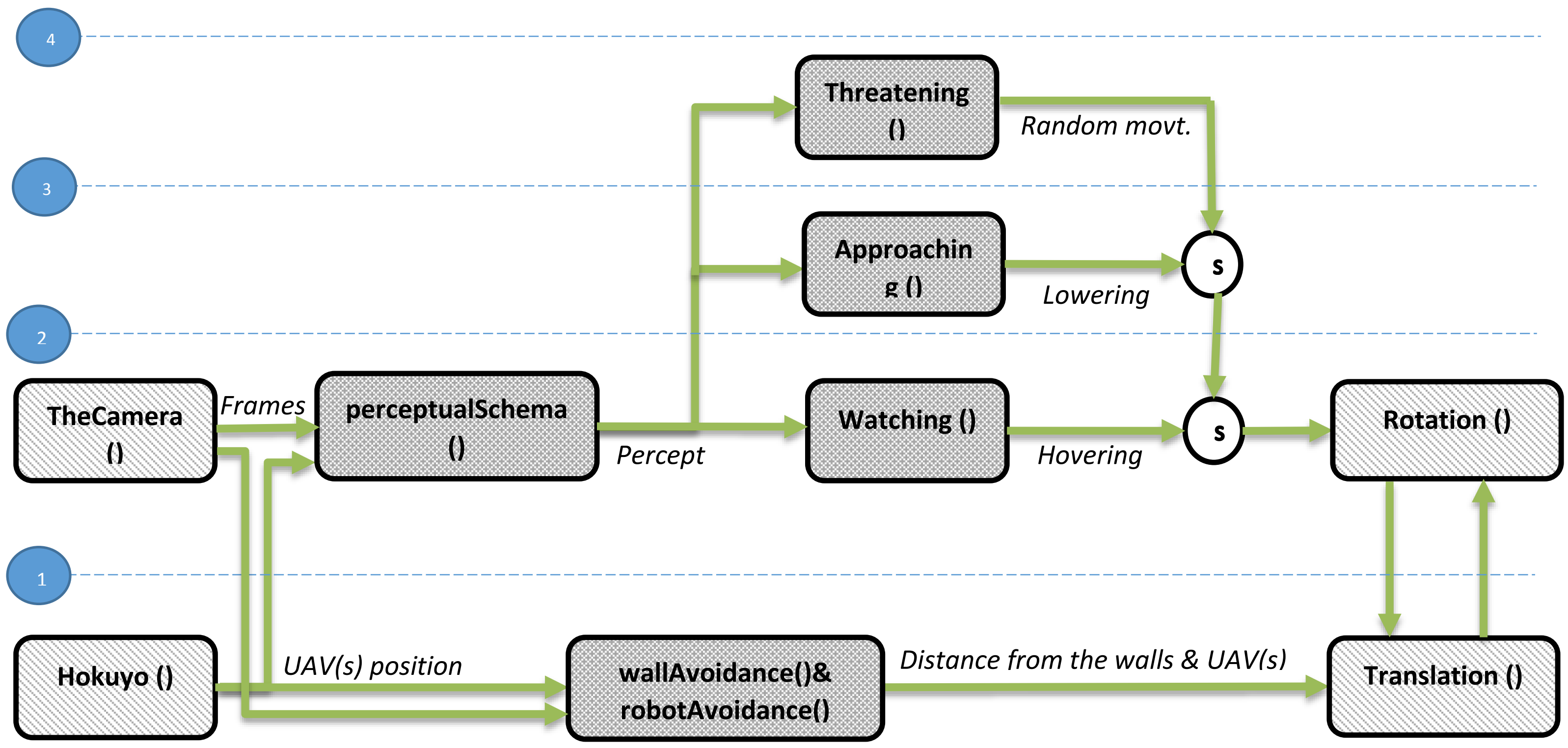


Figure 5 : Subsumption Design

Subsumption architecture has been employed to realize crowd tracking using UAV(s) and is as in the block diagram above.

The numbers in the diagram indicate the escalating set of behaviors which are,

1. wallAvoidance() & robotAvoidance(),
2. Watching() ,
3. Approaching() &
4. Threatening().

As can be seen from the diagram, the wallAvoidance() and robotAvoidance() behaviors are active all the time .Based on the percept , the UAV exhibits only one of the other three behaviors – watching , approaching or threatening .This is realized through suppression(**S**); suppressing the input going to a module (Rotation () in this case).This ensures that the UAV(s) not only tracks the crowd, but also maintains a safe distance from each other as well as the walls all the time.

MULTI-AGENTS

In the case of multi-agents, the robots are AWARE but not COORDINATED. This has been chosen for its simplicity, ease of design and more importantly effectiveness in performance.

In this scheme, the robots are kept away from each other with the use of potential fields. The number of robots needed for the task (blocking crowd) can be dynamically increased/decreased (with the press of 'j'/ 'k') .The region of operation of the robot is dynamically increased or decreased whenever each robot is retracted from or summoned into the scene respectively . This though, is done with some region overlap to ensure reliability in blocking the crowd. This aspect of design has helped achieve higher rate of success.

SOFTWARE ORGANISATION

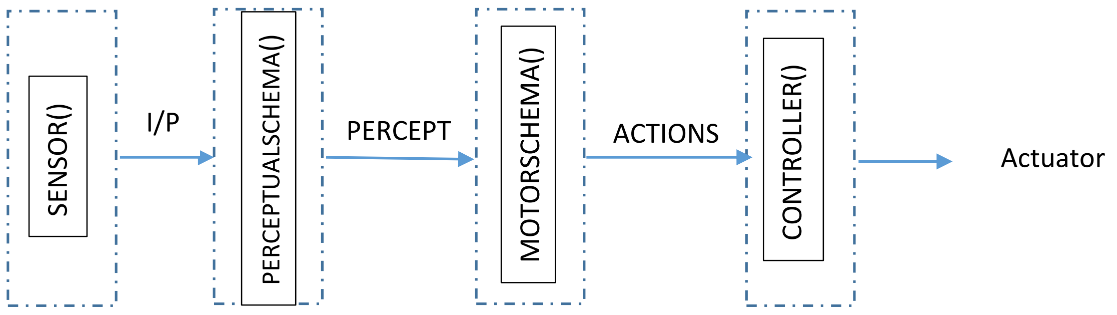


Figure 6 : Implementation Overview

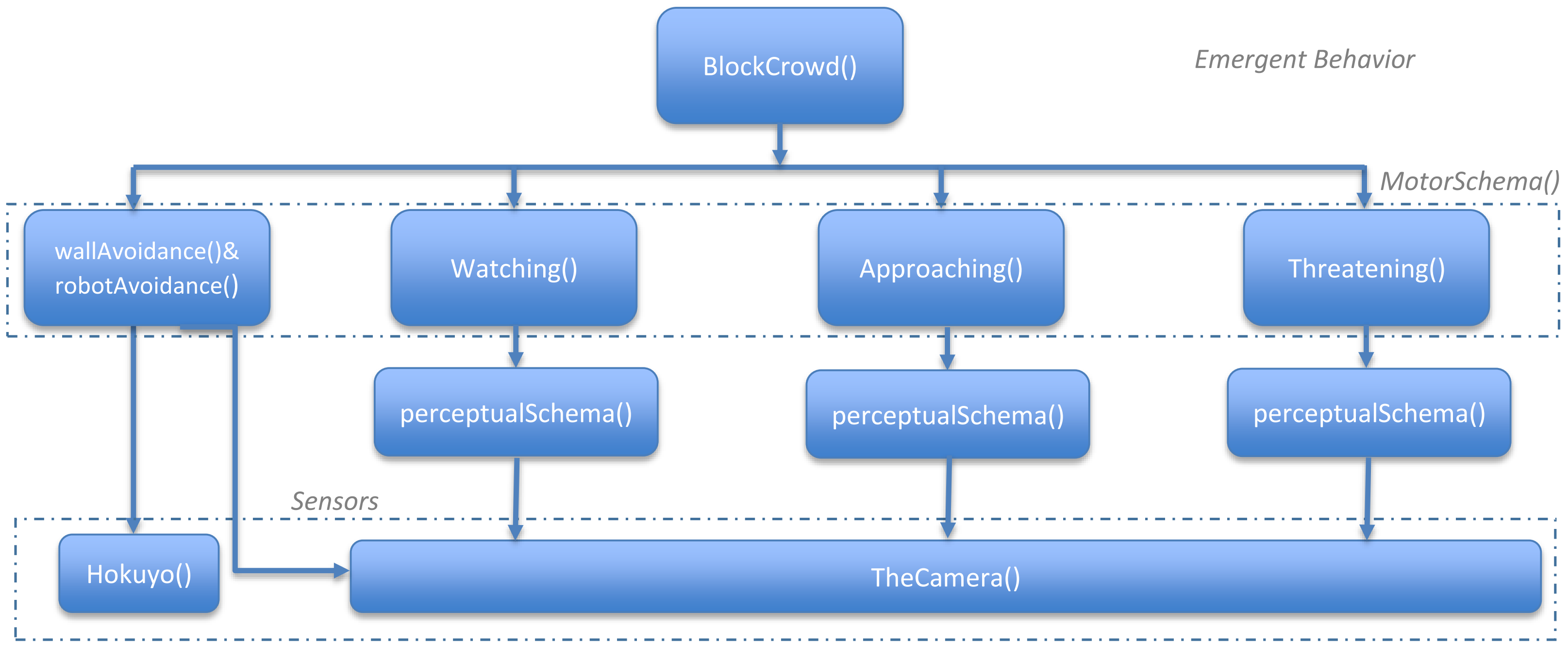


Figure 7: Detailed Implementation

For ease in mapping the design with implementation, a one-to-one correspondence between them has been established as can be observed from *Figure 7*, *Figure 8 & Figure 9.*

FUNCTION DESCRIPTION

The scripts and the functions defined or used in them are as detailed below. The purpose of this section is to familiarize the user with their purpose.

- **BlockCrowd**
  - This script is a placeholder serving to integrate the scripts associated with components in the subsumption architecture.
  - There are no functions declared in it but, has placeholder variables for perceptual schema, motor schema and sensors.
- **TheCamera**
  - This returns the people in the crowd.
- **CameraSwitching**
  - This is to provide different views of the scene.
  - There are three views possible ,
    - Main View (press ‘a’)
    - Reverse View (press ‘s’)
    - Overhead View (press ‘d’)
- **Hokuyo**
  - This provides the position of the UAV(s) and also helps to maintain repulsion between the UAVs, when two or more of them are summoned.
  - Function –
    - detectRobots() has been declared and used in this script to achieve the same.
- **PersonMove**
  - It is used to set the region for movement of the crowd, and to count the rate of success or failure in blocking the crowd.
- **Spawn**
  - This script has been adopted from an earlier work. See REFERENCES for more details.
  - Used to generate the crowd and the UAVs based on the user inputs.
  - There are three different modes for the crowd defined ,
    - Easy mode (press ‘1’), 1-2 people at any time.
    - Hard mode (press ‘2’), a dispersed crowd of 3 or more people.
    - Crazy mode (press ‘3’), a large dispersed crowd.
  - The UAVs (any number) can be summoned into the scene with each press of ‘j’ and retracted with the press of ‘k’.

- **PerceptualSchema**
  - This is to extract the needed information from the sensor inputs.
  - Function –
    - perceptualSchema(),
      - extracts the position information of the robot(s)
      - sets the target based on relative distance.
- **WallAvoidance**
  - It is used to keep the UAV(s) from running into the walls.
  - Function –

        - wallAvoidance(),determines the position of the robot relative to the walls based on the sensor inputs and prevents collision .
- **RobotAvoidance**
    - It is used to keep the robots from running into each other.
    - Function –
        - robotAvoidance(),determines the position of the other robots based on the sensor inputs and prevents collision.
- **MotorSchema**
    - It determines the state/behavior of the UAV(s) based on the inputs from the perceptual schema.
- **LowLevelController**
    - Decides the input to the actuators based on the selected state/behavior of the robot.
    - Functions –
        - control(),provides the control inputs to the actuator based on the selected state/behavior.
        - trackCenter(),determines the center of the closest target .
        - moveRandom2D(),provides the movement of the UAV while threatening.
        - findLine(),ensures the UAV finds the guard line (exit) and stays by it all the time while blocking the crowd.
        - stalk(),is to follow the crowd in the x-direction.
        - stabilize(),is to reset the position of the UAV(s) after approaching or threatening behaviors.
        - keepAlive(),defines the up and down movement of the UAV(s) during each state/behavior.

STRENGTHS & LIMITATIONS

- As can be observed from the simulations, the detection of people by the UAV is highly reliable. The performance is improved with increase in the number of UAVs.
- It is extremely quick and can retrace to its last valid position when forced out of the permitted region.
- The region of operation of each UAV is determined dynamically and there is a certain overlap between the regions to ensure that all the regions are well protected.
- This design is highly scalable; the user add and remove UAVs from operation dynamically.
- Since the sensor values are all simulated, noise and other errors in the input are not accounted for.
- Similarly, the effectiveness of the controller in getting work done through the actuators is yet to be justified.

REFERENCES

An attempt to refer to the works of other project teams which have been incorporated by has been made in this section.

1. The script Spawn has been adopted from the work of the team constituted by Wang & Zhou.
2. The idea of demarcating the region to show the crowd nearing the UAV has been adopted from the teams constituted by Nowak & Wilde and Ahmed, Hawkins & Probe.