Introduction to Game Programming and Robotics

Unit # 15

Acknowledgement

- Most of the examples/material presented in this presentation is taken from TinMan website maintained by Drew Noakes.

Installation Instructions

- http://simspark.sourceforge.net/wiki/index.php/Installation_on_Windows
  - MS Visual C++ 2008 Redistributable Package
  - Simspark
  - Rcssserver3d
  - Ruby (1.9.0, 1.9.1, 1.9.3) – make sure that your path variables are set appropriately
  - Update the SPARK_DIR and RCSSSERVER3D_DIR variables in rcssserver3d.cmd and other .cmd files as discussed on the website

Download TinMan and RoboViz

- TinMan
  - http://code.google.com/p/tin-man/downloads/list
- RoboViz
  - https://sites.google.com/site/umroboviz/

Agent Instantiation

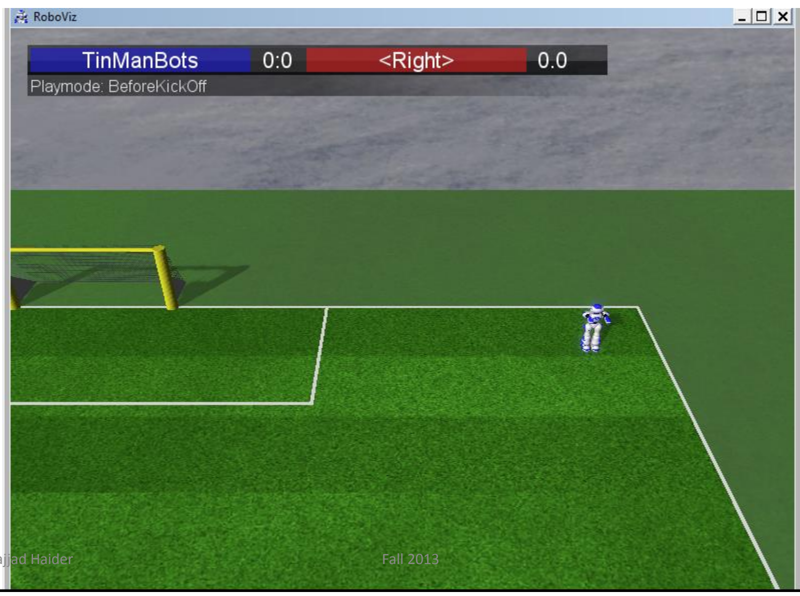
Original

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Program1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

After Modification

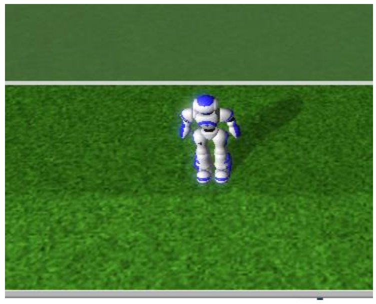
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using TinMan;

namespace Program1
{
    class Program : AgentBase<NaoBody>
    {
        public Program()
            : base(new NaoBody())
        {}

        public override void Think(PerceptorState state)
        {
            // TODO
        }

        static void Main(string[] args)
        {
            new AgentHost().Run(new Program());
        }
    }
}
```


Agent Instantiation (Cont'd)



Changing Position of Hands

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using TinMan;

namespace Program1
{
    class Program : AgentBase<NaoBody>
    {
        double gain;
        public Program()
            : base(new NaoBody())
        {
            gain = 2;
        }
        public override void Think(PerceptorState state)
        {
            // TODO
            Body.LAJ1.MoveToWithGain(Angle.FromRadians(-2.0), gain);
            Body.RAJ1.MoveToWithGain(Angle.FromRadians(-2.0), gain);
            Body.LAJ2.MoveToWithGain(Angle.FromRadians(0.35), gain);
            Body.RAJ2.MoveToWithGain(Angle.FromRadians(-0.35), gain);
            Body.LAJ3.MoveToWithGain(Angle.FromRadians(-1.4), gain);
            Body.RAJ3.MoveToWithGain(Angle.FromRadians(1.4), gain);
            Body.LAJ4.MoveToWithGain(Angle.FromRadians(-0.52), gain);
            Body.RAJ4.MoveToWithGain(Angle.FromRadians(0.52), gain);
        }

        static void Main(string[] args)
        {
            new AgentHost().Run(new Program());
        }
```

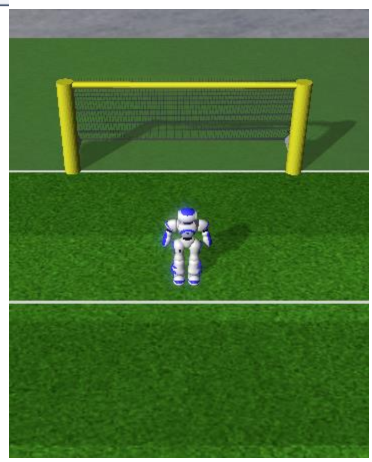


Sajjad Haider Fall 2013 7

Placing Agent at a Particular Position

Program1.Program | Program()

```
using TinMan;

namespace Program1
{
    class Program : AgentBase<NaoBody>
    {
        double gain;
        long counter;
        public Program()
            : base(new NaoBody())
        {
            gain = 2;
            counter = 1;
        }
        public override void Think(PerceptorState state)
        {
            if (counter == 5)
                Context.Beam(-9, 0, Angle.FromDegrees(0));

            // TODO
            Body.LAJ1.MoveToWithGain(Angle.FromRadians(-2.0), gain);
            Body.RAJ1.MoveToWithGain(Angle.FromRadians(-2.0), gain);
            Body.LAJ2.MoveToWithGain(Angle.FromRadians(0.35), gain);
            Body.RAJ2.MoveToWithGain(Angle.FromRadians(-0.35), gain);
            Body.LAJ3.MoveToWithGain(Angle.FromRadians(-1.4), gain);
            Body.RAJ3.MoveToWithGain(Angle.FromRadians(1.4), gain);
            Body.LAJ4.MoveToWithGain(Angle.FromRadians(-0.52), gain);
            Body.RAJ4.MoveToWithGain(Angle.FromRadians(0.52), gain);
            counter++;
        }
```



Sajjad Haider Fall 2013 8

Generating Hand Wave Movement

```
angles[0] = Body.LAJ1.Angle.Radians;
angles[1] = Body.LAJ2.Angle.Radians;
angles[2] = Body.RAJ1.Angle.Radians;
angles[3] = Body.RAJ2.Angle.Radians;

targets[0] = 0.5 * Math.PI;
 targets[1] = 0.25 * Math.PI * Math.Sin((t / 2) * 2 * Math.PI) + 0.25 * Math.PI;
 targets[2] = 0.5 * Math.PI;
 targets[3] = -0.25 * Math.PI * Math.Sin((t / 2) * 2 * Math.PI) - 0.25 * Math.PI;

for (int i = 0; i < 4; i++)
        velocities[i] = gain * (targets[i] - angles[i]);

 Body.LAJ1.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[0]);
 Body.LAJ2.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[1]);
 Body.RAJ1.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[2]);
 Body.RAJ2.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[3]);
```


Generating Hand Wave Movement (Cont'd)

```
namespace Program1
{
    class Program : AgentBase<NaoBody>
    {
        double gain;
        long counter;
        double[] angles = new double[4];
        double[] targets = new double[4];
        double[] velocities = new double[4];
        public Program()
            : base(new NaoBody())
        {
            gain = 2;
            counter = 1;
        }
        public override void Think(PerceptorState state)
        {
            if (counter == 5)
                Context.Beam(-9, 0, Angle.FromDegrees(0));

            double t = counter / 50;

            angles[0] = Body.LAJ1.Angle.Radians;
            angles[1] = Body.LAJ2.Angle.Radians;
            angles[2] = Body.RAJ1.Angle.Radians;
            angles[3] = Body.RAJ2.Angle.Radians;

            targets[0] = 0.5 * Math.PI;
            targets[1] = 0.25 * Math.PI * Math.Sin((t / 2) * 2 * Math.PI) + 0.25 * Math.PI;
            targets[2] = 0.5 * Math.PI;
            targets[3] = -0.25 * Math.PI * Math.Sin((t / 2) * 2 * Math.PI) - 0.25 * Math.PI;
```

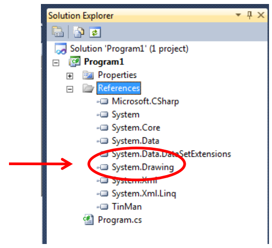
Generating Hand Wave Movement (Cont'd)

```
            for (int i = 0; i < 4; i++)
                velocities[i] = gain * (targets[i] - angles[i]);

            Body.LAJ1.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[0]);
            Body.LAJ2.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[1]);
            Body.RAJ1.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[2]);
            Body.RAJ2.DesiredSpeed = AngularSpeed.FromRadiansPerSecond(velocities[3]);

            counter++;
        }

        static void Main(string[] args)
        {
            new AgentHost().Run(new Program());
        }
    }
}
```


Agent's Position

- During a match, you need to find the position of your agent using localization.
- However, for the purpose of testing your localizer, the server can provide your agent's position.
- This is done by setting "setSenseMyPos" to true in the naoneckhead.rsg file
  - setSenseMyPos true

Using RoboViz



- In the top section, add
  - using TinMan.RoboViz;
- In the declaration section, add
  - RoboVizRemote roboViz;
  - Circle myCircle; // to draw a circle
  - FieldAnnotation myAnnotation; // for annotation

Using RoboViz (Cont'd)

```
public override void Think(PerceptorState state)
{
    double t = counter / 50;

    if (counter == 5)
    {
        Context.Beam(-9, 0, Angle.FromDegrees(0));
        roboViz = new RoboVizRemote(this);
        myCircle = new Circle { PixelThickness = 5, RadiusMetres = 0.5, Color = Color.Red };
        myCircle.CenterX = -9.0;
        myCircle.CenterY = 0;
        roboViz.Add(new ShapeSet("myCircle") {myCircle});
    }

    if (counter > 5)
    {
        if (state.AgentPosition != null)
        {
            myCircle.CenterX = state.AgentPosition.Value.X;
            myCircle.CenterY = state.AgentPosition.Value.Y ;
            myCircle.IsVisible = true;
        }
        else
            myCircle.IsVisible = false;
    }
    counter++;
}
```

Using RoboViz (Cont'd)

```
if (counter == 5)
{
    Context.Beam(-9, 0, Angle.FromDegrees(0));
    roboViz = new RoboVizRemote(this);
    myCircle = new Circle { PixelThickness = 5, RadiusMetres = 0.5, Color = Color.Red };
    myCircle.CenterX = -9.0;
    myCircle.CenterY = 0;
    roboViz.Add(new ShapeSet("myCircle") {myCircle});

    myAnnotation = new FieldAnnotation { Position = new Vector3(-9, 0, 2), Color = Color.White };
    myAnnotation.Text = "-9, 0";
    roboViz.Add(new ShapeSet("myAnnotation") {myAnnotation});
}

if (counter > 5)
{
    if (state.AgentPosition != null)
    {
        myCircle.CenterX = state.AgentPosition.Value.X;
        myCircle.CenterY = state.AgentPosition.Value.Y;
        myCircle.IsVisible = true;

        myAnnotation.X = state.AgentPosition.Value.X;
        myAnnotation.Y = state.AgentPosition.Value.Y;
        myAnnotation.Text = state.AgentPosition.Value.X + ", " + state.AgentPosition.Value.Y;
        myAnnotation.IsVisible = true;
    }
    else
    {
        myCircle.IsVisible = false;
        myAnnotation.IsVisible = false;
    }
}
```


RoboViz Control Keys

General

Input	Function
escape / q	quit RoboViz
F1	enter full-screen mode

Camera

Input	Function
lmb-drag	rotate camera in place
rmb-hold / page up	translate camera up
mwheel +	translate camera forward
mwheel -	translate camera backward
w / up arrow	translate camera forward (along field plane)
a / left arrow	translate camera left
s / down arrow	translate camera backward (along field plane)
d / right arrow	translate camera right
page down	translate camera down
1 - 7	sets camera to predefined positions
spacebar	enable ball tracking / automated camera mode

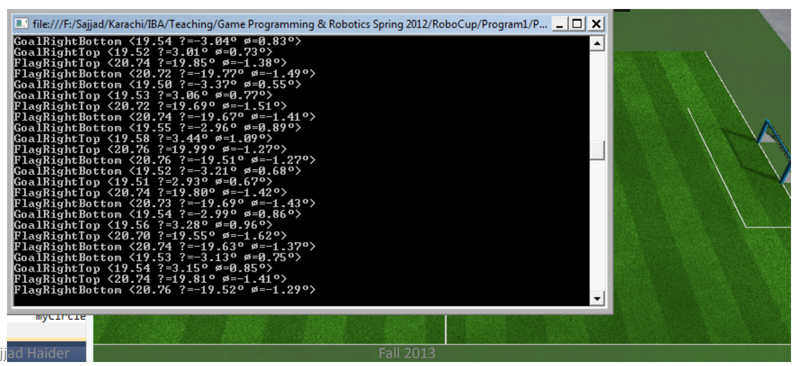
RoboViz Control Keys (Cont'd)

Live Mode

Input	Function
lmb-click	select object; deselects if no object is under cursor
ctrl-lmb-click	moves selected object to cursor position
o	display play-mode screen
up arrow	select previous item in list (play mode screen)
down arrow	select next item in list (play mode screen)
p	display drawings panel
t	toggle all drawings
f	display 2D field overlay
i	display robot player numbers
l	free kick left
r	free kick right
b	drop ball
k	kick off (left)
v	toggle robot perspective (when agent selected)

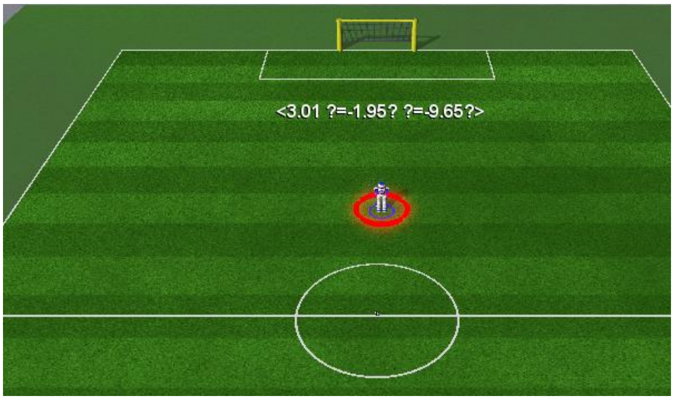
Accessing Landmark

```
if (state.LandmarkPositions != null)
{
    foreach (LandmarkPosition lMark in state.LandmarkPositions)
            Console.WriteLine(lMark.Landmark + " " +
                                    lMark.PolarPosition.ToString());
}
```



Accessing Ball Position

```
if (state.BallPosition != null)
        myAnnotation.Text = state.BallPosition.Value.ToString();
```



Localization using Triangulation

- RoboCup 3D server provides us the landmark data in the following form
  - <distance, theta, phi>
- If we get information about at least two landmarks then we can localize ourselves by solving two equations simultaneously.

Landmarks Positions

- We already know the landmark labels and corresponding locations
  - FlagRightTop (10.5, 7)
  - FlagRightBottom (10.5, -7)
  - GoalRightTop (10.5, 1.05)
  - GoalRightBottom (10.5, -1.05)
  - FlagLeftTop (-10.5, 7)
  - FlagLeftBottom (-10.5, -7)
  - GoalLeftTop (-10.5, 1.05)
  - GoalLeftBottom (-10.5, -1.05)

Example: Finding Location

- Suppose we get the following two observations
  - FlagRightTop: <17.94, 25.54, 24.74>
  - FlagRightBottom: <17.90, -25.65, 24.49>
- Assume that we are standing at (x, y).
- Using the above data and knowing the position of FlagRightTop and FlagRightBottom {(10.5, 7) and (10.5, -7)}, we can form two equations as follows:
  - $17.94^2 = (x - 10.5)^2 + (y - 7)^2$
  - $17.90^2 = (x - 10.5)^2 + (y + 7)^2$

Example: Finding Orientation

- We get our location (x, y) by solving these two equations.
- The next task is to find our orientation.
- We know that angle between two points can be computed as
  - $\tan(\text{Orientation}) = (y1 - y) / (x1 - x)$
- In this case, (x, y) is our location and (x1, y1) can be the location of any of the observed landmark.

Example: Finding Orientation (Cont'd)

- We can rewrite the tangent equation as
  - $\text{Orientation} = \tan^{-1}((y - y1) / (x - x1))$
- But since we are already provided Θ by the server, we need to subtract it as well.
- The revised equation, thus becomes
  - $\text{Orientation} = \tan^{-1}((y - y1) / (x - x1)) - \Theta$

Example: Finding Orientation (Cont'd)

- For the current example, if we are using FlagRightBottom data (<17.90, -25.65, 24.49>) and assuming that our own location was found to be at (-6, 0), then the equation becomes
  - Orientation = $\tan^{-1}((0 + 7) / (-6 – 10.5)) – (-25.65)$
- – Note that the first part of the equation is in radians while the second part is in degrees. So first convert -25.65 into radians and then subtract it.

Combining Localization using Different Landmarks

- The calculations of the previous slides help us in finding our location and orientation using two landmarks.
- During a match, it happens quite often that we observe more than two landmarks.
- In this case, we perform the same process for each pair of landmark and then average all the locations (x and y coordinates) and orientations.
- This helps in reducing the noise factor.

Adjustment for Head Movement

- The calculations showed in the previous slides assume that our head's joints (HJ1 and HJ2) were at 0 degree.
- In case, we were moving our head then we need to adjust the observed values accordingly.