

Angry Birds Basic Game Playing Software

version1.32

By

XiaoYu (Gary) Ge, Stephen Gould, Jochen Renz

Sahan Abeyasinghe, Jim Keys, Andrew Wang, Peng Zhang

Research School of Computer Science

The Australian National University

2014-10-20

Contents

<i>Install JAVA environment.....</i>	<i>3</i>
<i>Install Chrome</i>	<i>3</i>
<i>Install the Software</i>	<i>3</i>
<i>Install Chrome Plugin.....</i>	<i>3</i>
<i>Run the Software.....</i>	<i>4</i>
<i>Angry Birds Extension.....</i>	<i>5</i>
<i>AI Agent.....</i>	<i>5</i>
<i>Proxy.....</i>	<i>5</i>
<i>S/C Communicating Port.....</i>	<i>5</i>
<i>Vision Module</i>	<i>5</i>
Obtaining MBRs	6
Obtaining Real Shape	6
<i>Trajectory Module.....</i>	<i>7</i>
<i>Naive Agent</i>	<i>8</i>
<i>Play the Naive Agent</i>	<i>9</i>
Load the Game Window	9
Run the Naive Agent.....	9
<i>Create your own Intelligent Agent</i>	<i>10</i>
Object Representation	10
Screen Shot and Segment an image	11
Access the Game State	11
Trajectory Module.....	11
Execute Shots.....	12
Self-Manage mode.....	12
Auto-Arrange mode	12
Other Useful Methods	13
Get the type of the bird on the sling.....	13
Get all the supports of an object	13
Reachability.....	13
Compile using ANT	14

Installation

Install JAVA environment

The framework has been tested with the JAVA6 and above. To check your java version, you can enter `java -version` in your command window.

You can download java environment (JRE) from this link

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>




Install Chrome

Chrome can be obtained from <http://www.google.com/intl/en/chrome/browser/>

Install the Software

Unzip the software package, assuming the directory of the software folder is `./angrybirds/`

You will have the following files/folders:

-  The `external` folder contains all necessary libraries.
-  The `src` folder contains all source files
-  The `plugin` folder: extension for interfacing with Chrome.

Install Chrome Plugin

1. Open Chrome
2. Go to: `chrome://chrome/extensions/`
3. Click the 'Developer mode' check box
4. Click the 'Load unpacked extension...' button
5. Browse for the 'plugin' folder (contains the 'manifest.json' file)

NOTE: if the screenshots are coming out as transparent PNG files do this:

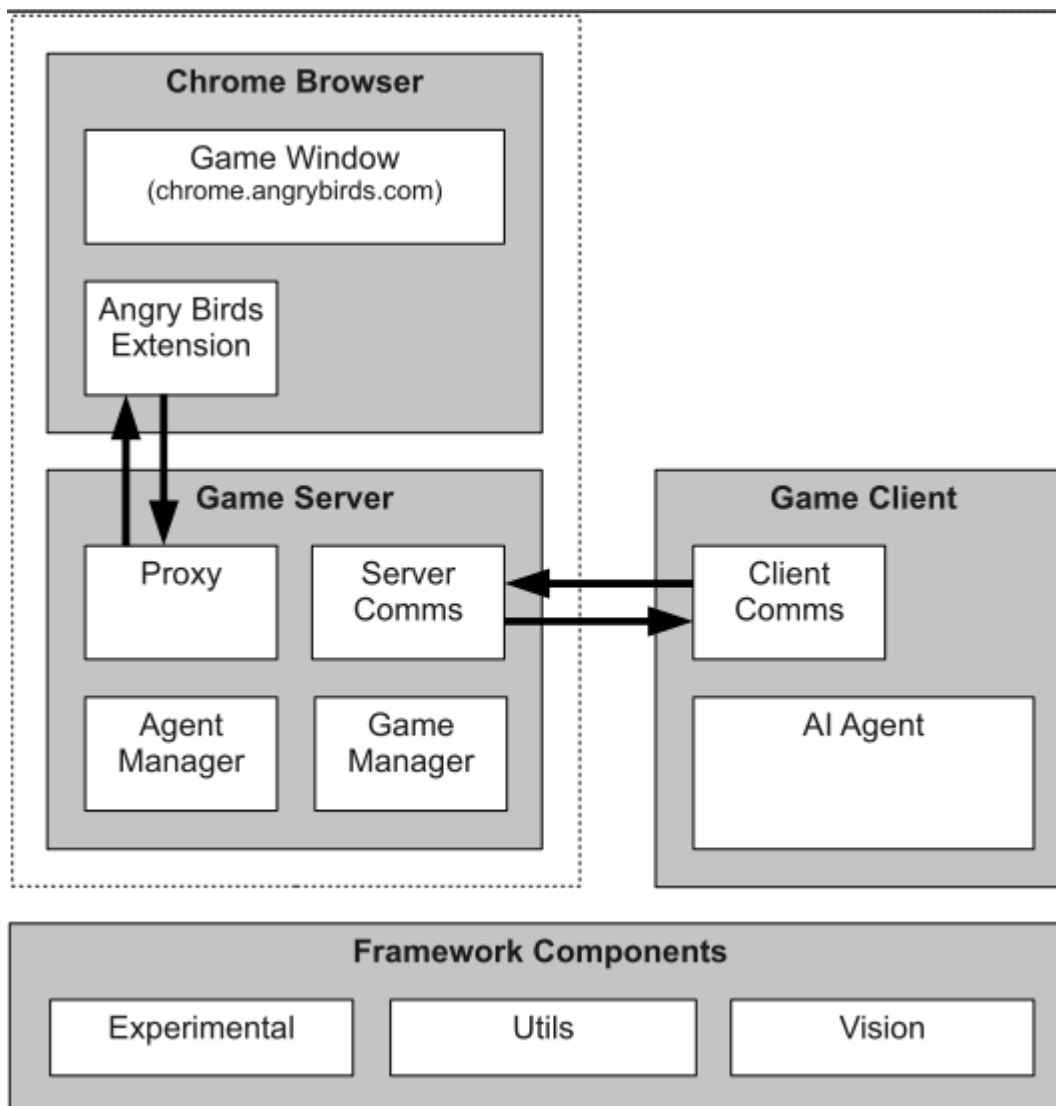
1. Open Chrome
2. Go to: `chrome://flags/`

3. Find the option 'Disable WebGL' and enable it (i.e. enable the disabling of WebGL)

Run the Software

Through the command line, you can run the naive agent (see section [Run the Naive Agent](#)), view the real time segmentation of the game images (see section [Vision Module](#)) and view the real-time trajectory output (see section [Trajectory Module](#)).

Server-Client Architecture



Angry Birds Extension





The chrome plugin interacts with the chrome browser. It offers functionalities such as capturing game window, executing actions (e.g. move the mouse, click, and wheel to zoom). To install it, please see Install Chrome Plugin

AI Agent

An agent can use the vision module to analyze game scenarios and the trajectory module to plan on shots. Although the framework is written in java, the agent can be implemented in other languages such as c++, python, etc.

Proxy





The server interacts with the angry birds extension via the proxy module. The following are the proxy messages:

-  CLICK: left click of the mouse
-  DRAG: drag the cursor from one place to another
-  MOUSEWHEEL: scroll the mouse wheel
-  SCREENSHOT: capture the current game window

The agent does not need to access the Proxy component.

S/C Communicating Port

Server/Client Communicating Port receives messages from agents and sends back feedbacks after the server executed the required actions indicated by the messages. The messages fall into the following three categories

-  Configuration messages
-  Query messages
-  In-Game action messages
-  Level selection messages

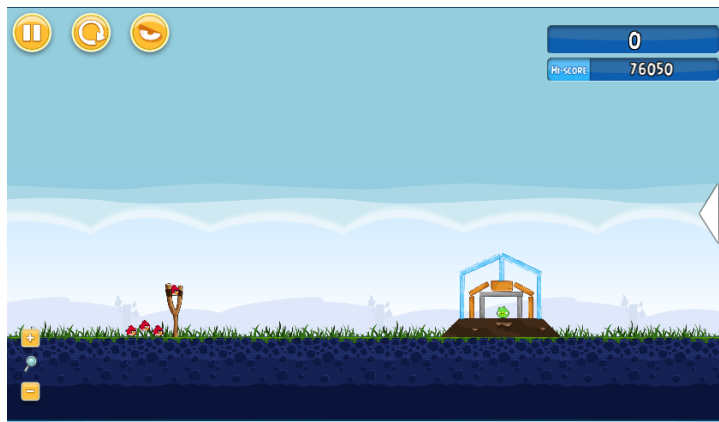
The syntax of the messages can be found in /doc/ServerClientProtocols.pdf

Vision Module

The vision module is composed of two image segmentation components. One component segments an image and outputs a list of the minimum bounding rectangles (MBR) of essential objects in the image. The essential objects includes {"Sling", "Red Bird", "Yellow Bird", "Blue Bird", "Black Bird", "White Bird", "Pig", "Ice", "Wood", "Stone", "TNT", "TrajPoints"}. The other component outputs real shapes instead. You can use both of them to process a screenshot.

Obtaining MBRs

It takes about 100 ms to compute the MBRs of a typical scenario. The following is an example of the segmenting result (MBRs).



To show the real-time segmentation, you can type and execute the command in the command window under the software directory.

```
java -jar ABSoftware.jar --showMBR
```

Obtaining Real Shape

It takes about 300 – 500 ms to compute the real shapes. In addition to the essential objects, this image segmentation can recognize hills and the ground. Hills are those dark brown obstacles that will not be affected or destroyed by other game objects. The following is an example of the segmenting result (Real Shape).



To show the real-time segmentation, you can type and execute the command in the command window under the software directory.

java -jar ABSoftware.jar --showReal

Trajectory Module

The Trajectory Module estimates the trajectory that a bird will follow given a particular release point (relative to the slingshot). We use a constant velocity for every shot within the same level and use the release point and slingshot location to determine the launch angle α . From these we can compute the initial horizontal and vertical velocities as $v \cos(\alpha)$ and $v \sin(\alpha)$, respectively.

With the initial velocities in hand, we use Newton's classical laws of physics to estimate the parabolic path that the bird will take. The predictTrajectory function in the Trajectory Module returns a list of points that the bird will follow given a certain release point. To facilitate planning shots, the Trajectory Module provides a function (called estimateLaunchPoint) to estimate the release point given a desired target (for example, the centre of one of the detected pigs). The function uses the following equation to calculate the launch angle

$$\theta = \arctan\left(\frac{v^2 \pm \sqrt{v^4 - g(gx^2 + 2yv^2)}}{gx}\right)$$

Where (x, y) are the normalised coordinates of the target point relative to the sling (using sling size as the scale), and the gravity g is assumed to be 1 unit. Two angles are obtained from this equation and the function returns two corresponding launch points in an ArrayList of Points.

Note that when a shot is made, the actual launch angle α is always slightly different from the arctangent of the launch vector and the difference varies from level to level. Also, the normalised

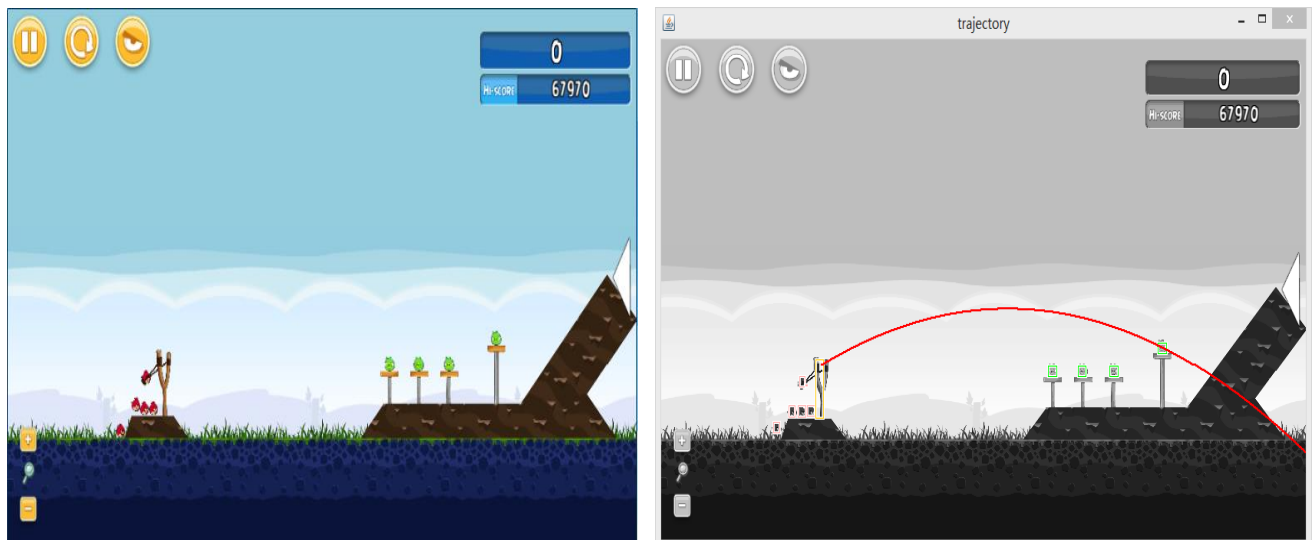
velocities are slightly different between levels. The variation in angle and velocity are taken into account when predicting trajectories and estimating launch points. The trajectory module provides a method (`adjustTrajectory`) to adjust these two changes using information from a previous shot, so that the accuracy of the following shots is improved.

Also note that due to pixel level variations in the location of the slingshot and bird and effects due to scaling, the paths estimated by the Trajectory Module are only approximate. Agents should take this into account when planning their shots.

To show the real-time segmentation, you can type and execute the command in the command window under the software directory.

```
java -jar ABSoftware.jar -showTraj
```

The following is an example output



The Naive Agent (standalone) – a demo

Naive Agent

The naive agent is created to demonstrate how to construct an agent on the basis of the provided modules, namely, the chrome plugin, the vision module, and the trajectory module. The naive agent is called “naive” because it shoots the bird directly to the pig without reasoning.

The code can be found in `/src/ab/demo/NaiveAgent.java`

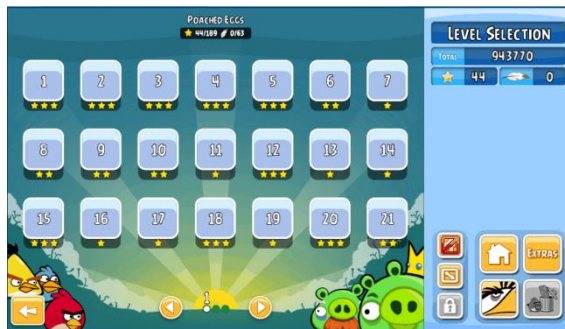
This work is licensed under the terms of the GNU Affero General Public License as published by the Free Software Foundation and can be accessed via <http://www.aibirds.org/basic-game-playing-software.html>

Play the Naive Agent

Load the Game Window

To load the game window

1. Open Chrome
2. Maximize the browser window
3. enter "<http://chrome.angrybirds.com/>" in the address bar, then press enter
4. Select **SD** version, then click Play
5. Select the episode **POACHED EGGS** and stay at the level selection page.



(The level selection page)

Please keep the tab of the game activated. You do not need to keep the browser window in front of the screen. I.e. you can minimize the window after loading the game, but ensure the tab remains selected.

Run the Naive Agent

To run the naive agent

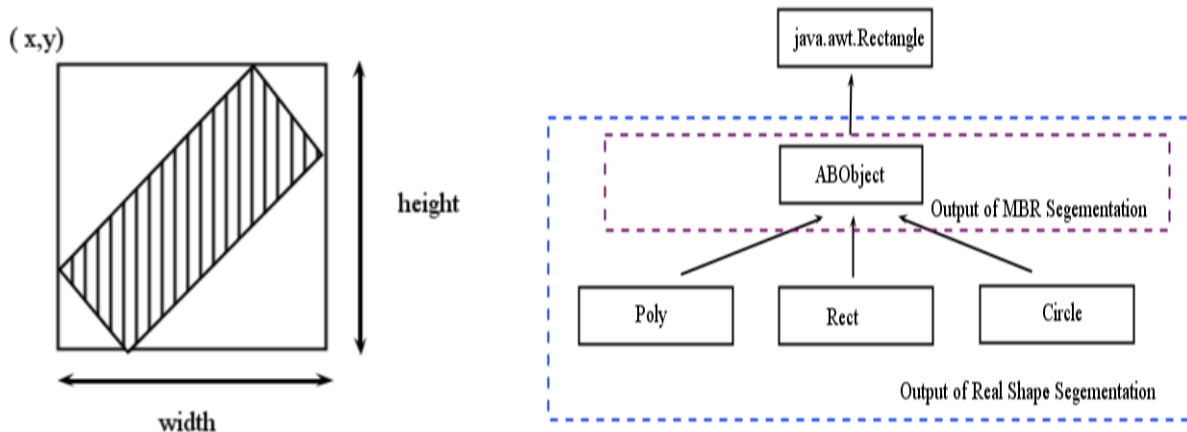
1. Load the game in Chrome (see section [Load the Game Window](#))
2. Open the command window.
3. Go to the software directory.
4. Execute the command `java -jar ABSoftware.jar -na [1-21]` to invoke the naive agent, the naive agent will start from the level you specified. The initial level is 1 by default.

E.g. `java -jar ABSoftware.jar -na 2` will start the naive agent in the level 2. Note: the agent cannot access a level before unlocking it.

Create your own Intelligent Agent

Object Representation

For each object in a screenshot, the vision module will create a java object over it. The java object is specified in ABOBJECT.java that extends java.awt.Rectangle. The ABOBJECT class has four main attributes, namely x, y coordinate of the top left corner, width and height.

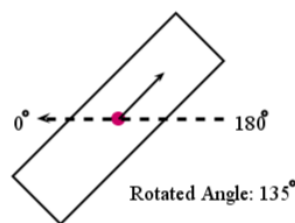
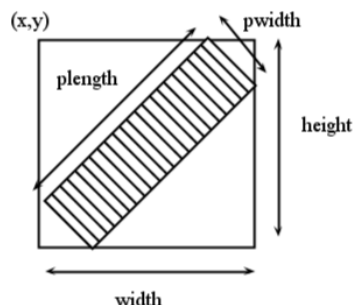


The ABOBJECT class has an attribute indicating the object type of a game object. The object types are: {"Hill", "Sling", "Red Bird", "Yellow Bird", "Blue Bird", "Black Bird", "White Bird", "Pig", "Ice", "Wood", "Stone", "TNT"}.

Attribute "hollow" indicates whether an object is hollow (has a hole in the middle) or not. Note: The MBR segmentation does not distinguish between hollow and solid objects.

We classify the shapes of the game objects into 3 groups, namely Circle, Rect, and Poly (refer to /src/ab/vision/real/shape). There is a corresponding java class for each group, and the java classes extend ABOBJECT.java so that they keep all the methods and attributes of ABOBJECT.

- Circle: represents a circular shape by the circle centre and radius.
- Rect: represents a rectangle of an arbitrary angle. In addition to the four attributes which specifies a MBR, the Rect class represents the real shape of a rectangle by introducing three more attributes: **pwidth**, **plength**, and **angle**. The pwidth, plength maintains the length of the shortest and longest edge respectively.



- Poly: represents a polygon using java.awt.Polygon. Triangles are represented by Poly.

Screen Shot and Segment an image

It is very easy to take a screen shot and segment it. The following is a code fragment showing the process.

```
//Start the game proxy
ActionRobot aRobot = new ActionRobot();
//Capture an Image
BufferedImage screenshot = aRobot.doScreenShot();
// Initialize the vision component
Vision vision = new Vision(screenshot);
// Get a list of MBRs of all the pigs in the screenshot
List<ABObject> pigsMBR = vision.findPigsMBR();
// Get a list of real shapes of all the pigs in the screenshot
List<ABObject> pigsReal = vision.findPigsRealShape();
```

As can be seen from the code, we provide a macro vision (see `/src/ab/vision/Vision.java`) that manages the two image segmentation components. The method `findPigsRealShape()` uses the real-shape segmentation to obtain a list of pigs while `findPigsMBR()` uses the MBR segmentation. Neither of the segmentation components recognizes all the object categories. The MBR segmentation cannot detect the hill while the real-shape segmentation cannot recognize the TNT. So `findTNTs()` uses the MBR segmentation, and `findHills()` uses the real-shape segmentation.

Access the Game State

There are four main game states

1. "WON": a page shows the final grade and the stars you won.
2. "LOST": a big pig tells you lost
3. "PLAYING": the game is not finished
4. "LEVEL SELECTION": the level selection page.
5. "LOADING": the game is currently loading and a progress bar is showing on the page

Please refer to `/src/ab/vision/GameStateExtractor.java` for details

Trajectory Module

Once you know your target, you need the trajectory module to plan on the shots. To get the set of estimated release points given a target, you can call the `estimateLaunchPoint(Rectangle, Point)` method. This method requires the MBR of the slingshot and the target point

```
//Initial a trajectory planner
TrajectoryPlanner tp = new TrajectoryPlanner();
//Get the estimated points
List<Point> pts = tp.estimateLaunchPoint(sling, target);
```

Please refer to `/src/ab/planner/TrajectoryPlanner.java` for details

Execute Shots

A Shot contains six parameters (x, y, dx, dy, t_shot, t_tap) The first four parameters specify a mouse track from (x,y) to (x + dx, y + dy). T_shot is the time at which a bird is supposed to be launched. T_tap specifies the gap between the releasing time and the corresponding tapping time. Time are counted in milliseconds. You can submit a list of shots to the server and the server will execute shooting accordingly.

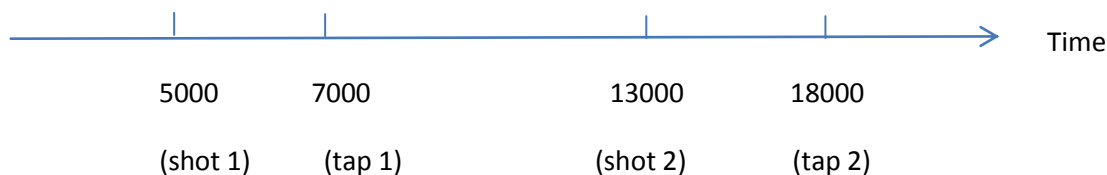
More details about executing shots are provided in the section **In Game Action Messages** of *ServerClientProtocols.pdf* (Can be found under the doc directory).

Self-Manage mode

Let's say you want to shoot 2 birds and the following are corresponding shot instances:

- 1) Shot (x,y,dx,dy, 5000,2000);
- 2) Shot (x,y,dx,dy, 13000,5000);

You can pass the two shots to the server, and the server will shoot the bird following the timeline below:



Note: t_tap specifies the gap between the releasing the time and the corresponding tapping time. When you specify the shooting time, please beware that:

- 1) Avoid executing a shoot and tap actions at the same time. E.g. The following are two conflicting shots:

Shot (x,y,dx,dy, 5000,3000); Shot (x,y,dx,dy, 8000,5000);

- 2) In the game, there is a gap between two shots. You cannot shoot a bird immediately after one shot. The gap is around 5 seconds. So please adjust your t_shot and ensure the gap between two shots is more than 5000.
- 3) The tapping time of the first shot cannot be after the shooting time of the second shot.

Auto-Arrange mode

We provide another convenient way for shooting, if you do not care about the exact time. We can arrange the shooting time for you. By setting the t_shot field of each shot in the shot sequence to 0, the server will shoot the bird every 5 seconds. For example,

- 1) Shot (x,y,dx,dy, 0,3000);
- 2) Shot (x,y,dx,dy, 0,5000);

The first shot will be made at time 0, and second shot will be made at time 5000. Besides, the first tap happens at time $3000 + 0 = 3000$ and the second tap occurs at time $5000 + 5000 = 10000$.

Note, if you want the server to arrange the shooting for you, you need to ensure **ALL** the `t_shot` fields are set to 0 in a shooting sequence.

The server will treat the following conflicting shooting sequence in the self-manage mode

- 1) Shot (x, y,dx,dy, 0,3000);
- 2) Shot (x,y,dx,dy, 0,5000);
- 3) Shot (x,y,dx,dy, 10000,5000);

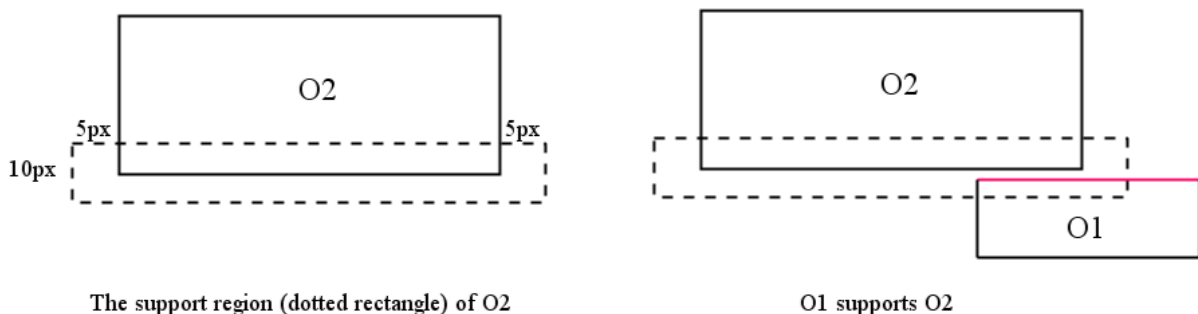
Other Useful Methods

Get the type of the bird on the sling

You can get the type of the bird on the sling by using the method `getBirdTypeOnSling()` (see `/src/ab/demo/other/ActionRobot.java`). This method will first zoom in on the sling, take a screenshot, and segment the screenshot to get a list of birds. The method then sorts the list to find the highest bird, and returns the type of that bird.

Get all the supports of an object

The method `isSupport(ABObject o1, ABObject o2)` (see `/src/ab/utils/ABUtil.java`) will return true if o1 supports o2. The method determines the support relation by checking whether the top edge of the MBR of o1 is within the support region (depicted as the dotted rectangle in the following figure) of the MBR of o2. The support region is the rectangle with the (x, y, width, height) as $(o2.x - \text{gap}, o2.y + o2.height - \text{gap}, \text{gap} * 2 + o2.width, \text{gap} * 2)$. The gap is adjustable. The gap is 5px in the figure.



The method `getSupporters(ABObject o2, List<ABObject> objs)` returns a list containing the subset of objs that support o2.

As you notice, this method cannot deal with the support of angular rectangles. You may want to develop your own strategy to perform a comprehensive supports check.

Reachability

The method `isReachable(Vision vision, Point target, Shot shot)` (see `/src/ab/utils/ABUtil.java`) returns true if the point (target) is reachable (without obstruction) by the shot.

The method first obtains a list of all the trajectory points of the shot, and retains those that are at the left of the target. I.e. the x coordinate of the point is smaller than the target's. Then the method checks whether any of the points is contained in the MBR of any blocks (wood, ice, and stone), or

has a similar RGB value to the color of hill. The method returns true (reachable) when no such points exist.

You may want to develop your own strategy to determine the reachability.

Compile using ANT

You can use ANT to compile the source codes via command line. Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. For more information, please refer to <http://ant.apache.org/>

You can download ANT from <http://ant.apache.org/manualdownload.cgi>

ANT 1.7 or above required

To compile the source codes, go to the software directory, and use command

```
ant compile
```

To generate an executable file

```
ant jar
```