Being in the Game: Implementing First Person

Adding Challenge to Journey or PacMan by switching viewpoints from bird’s eye to first person so that the player only sees what the Traveler or PacMan sees rather than the entire game world.

Created by: Catharine Brand, University of Colorado

This curriculum has been designed as part of the Scalable Games Design project. It uses material from Fred Gluck’s video tutorials and Susan Miller’s SGD curricula.

This material is based upon work supported by the National Science Foundation under Grant No. DRL-1312129 and CNS-1138526. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Lesson Objective:

• To add rules that let the player see the game from the point of view of the key-controlled character.

Prerequisite Skills:

• Students are presumed to have the following skills. Return to the Frogger Lesson Plans for detailed explanations on these skills.
  o Create agents
  o Basic agent behavior including:
    ▪ Key-controlled movement
    ▪ Random movement
    ▪ Ending the game

Computational Thinking Patterns:

• First Person

Useful links:

first person video tutorial:
http://sgd.cs.colorado.edu/wiki/First_Person_Navigation
introduction to modulo math operation:
http://betterexplained.com/articles/fun-with-modular-arithmetic/

Activity Description: divide over 2 to 3 classes

• Part 1: Create a basic world with a key-controlled ladybug and 4 walls.
• Part 2: Learn how to name directions in AgentCubes. Initialize the ladybug’s Direction attribute and modify the basic movement rules so that the ladybug updates her Direction attribute to match the direction she is moving and then turns to face the same direction.
• Part 3: Explore why movement should be different in first person: up arrow means move forward and other arrows turn agent. Learn what mod means and why it is important for first person movement.
• Part 4: Add the first person movement rules.
• Part 5: Test the first person movement rules.
• Part 6: Copy and paste first person rules into the Traveler or Pacman.
Table of Contents

Teacher Instructions:  Being in the Game –
Implementing First Person Navigation

Student Handout:  Being in the Game –
(under development)  Implementing First Person Navigation

Vocabulary/Definitions

Agent Attribute ..........a value assigned to an agent (such as scent)
Bird’s Eye View....... looking down on a World as if you were flying over it
First Person View....seeing a world from the point of view of a single agent
Local Variable.........a variable (attribute) belonging to a specific agent
Mod or Modulo ……this mathematical operation returns the remainder when one number is divided by another

General Teaching Strategies

Basic Philosophy

- The educational goal of these lessons is to learn and apply Computational Thinking Patterns in the context of a familiar game. Emphasis on these Computational Thinking Patterns is essential for student understanding.

- Every effort has been made to create instructions with an eye toward guided discovery. Direct instruction has been used for those aspects where students are learning the code for the first time; however, materials have been provided to ensure that students are

1This information is supported by research found in the following documents:
Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010, June). Using scalable game design to teach computer science from middle school to graduate school. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 224-228). ACM.
understanding the programming concepts, as opposed to simply copying code. Note that special materials have been designed for students who are new to AgentCubes.

- Student materials are available for each portion of the game design. These materials are intended to be used in addition to teacher materials, which provide prompts and discussion points. Students may become frustrated with too little teacher support. Students may lose out on conceptual understanding with too much teacher support.

Guided Discovery Process

- **Model the process** rather than just giving students the answer. Building the game on your own, before trying it with your class will enable you to see which steps may challenge or confuse your students.

- Have students work through problems independently. Ask directing questions or give helpful suggestions, but **provide only minimal assistance** and only when needed to overcome obstacles.

- Don’t fear **group work**! It is common for computer programmers to talk through problems with one another, and to use code snippets found from other programs, and other programmers. Talking through coding problems enables students to think more critically about Computational Thinking Patterns, as well as the steps needed to solve a problem. Additionally, seeing how others solved an issue with code helps students realize that problems often have multiple solution strategies, and that some solutions might be more effective than others.

- Recognize that programming is largely a process of **trial and error**, particularly when first learning. It is helpful to encourage this mindset with your students.

Building Blocks

- Each project is designed to build on the prior one. Very little student support is provided where expertise has already been created. Conversely, material that is new has more support.

- Be sure to talk through the building blocks (especially for PacMan in the area of diffusion and hill climbing) as these Computational Thinking Patterns will appear often in future games and simulations.

- Remember that conceptual understanding takes time, and it may be necessary to explain these concepts multiple times, using different examples, so that all students can be successful.
Support Learning

- Research shows that game design is associated with engaged students, and engaged students show higher levels on conceptual understanding. Allowing students to personalize their games aids in this engagement and motivation.

- Coding may be difficult for some students, and all students are likely to be frustrated at times when the code does not produce the expected results. **Praise students** for sticking with the troubleshooting process and encourage them to share what they learned with others.

- Be sure to communicate that the process is more important than the answer, and that coding of a project often takes time. Do not place pressure on your students to ‘hurry up’ and resort to giving them the code. The process of figuring it out on his/her own will result in much stronger conceptual understanding.

Differentiated Instruction

*Note that there are many vocabulary words in this lesson that may be new for your students. Take time to define those words. Using the words in context often will reinforce their meaning for the students.*

- **Students who need a challenge**: Some students with more fluency in programming may finish this very quickly – be prepared for them to move on earlier than other students by having student materials ready in advance.

- **Students who need more assistance**: Other students may struggle a bit more.
  - This lesson provides an opportunity to review how degrees match positions on the unit circle and the meaning of the modulo operation.
  - Pairing the student with an experienced student should alleviate many problems.
  - Vocabulary for ELL Students: bird’s eye view, first person view, navigation, modulo, rotate
Teacher Instructions: Introduction

Background:
Up to this point, we have played games in *bird’s eye view*, looking down at the game world from above. From the bird’s eye view, it is easy to see how to get around obstacles, and where the dangers and the rewards of a game are located. Below is a journey world in bird’s eye view.

Gamers usually play games in first person because it can be exciting and challenging to make your way through the game world, seeing only what your character would see. Instead of looking down on an entire maze, the player sees only the walls that surround his character. Chasers jump out suddenly from behind a wall. A treasure appears unexpectedly as the gamer turns the corner. Below is a snapshot of the same world with all the agents in the same positions in *first person view*, looking at a Chaser in the maze over the Traveler’s head. Note that the direction which appears to be “up” in first person is actually “down” in bird’s eye view!

While it is easy to switch into the first person point of view by selecting an agent with the big arrow tool and clicking on the camera button located on the top bar of the AgentCubes window, the simple movement rules with arrow keys do not seem quite right. In these simple rules, the up arrow key always moves the character towards the top of the AgentCubes window although the direction that appears to be “up” in first person view may actually be the left, right or lower side of the AgentCubes window. The right and left arrow keys make the agent move sideways like a crab and the down key makes the agent move backwards. It would be more realistic if the left, right and down arrow keys turned the agent while the up arrow key moved the agent forward in whichever way it is facing. In order to make the agent’s movements match what the player sees from the first person perspective, we must create new movement rules for our keyboard-controlled agent when the first person button has been clicked.
Creating a simple world for exploring first person movement
Testing our rules is easier if we work in a simplified world where we can instantly see which way our agent is facing. We will add the first person rules to a ladybug because it has a head end, which makes it much easier to see that it turns and moves correctly in response to each arrow key.

Make a new level called First Person in the game in which you would like to use the first person perspective.
Select PacMan or the Traveler or Frogger or any main character who is keyboard-controlled with the big arrow tool, click on the +Shape button to make a 2nd shape and name it ladybug.
Choose Inflatable Icon, the Animals category and then Akako, the ladybug.
Use this shape for this lesson or create a new agent named Ladybug and choose the akako shape.

Create the following 5 agents:
1. a tile named “Ground”
2. Select a cube for the north wall by choosing the Letters-Bit font category and then picking n.png.
3. Select a cube for the west wall by choosing the Letters-Bit font category and then picking w.png.
4. Select a cube for the south wall by choosing the Letters-Bit font category and then picking s.png.
5. Select a cube for the east wall by choosing the Letters-Bit font category and then picking e.png.

Create your first person world:
1. Cover it with a layer of Ground tiles.
2. Put the Ladybug in the center of the World with her head facing the top of the World.
3. Draw a line of north blocks across the top of the World.
4. Draw a line of west blocks down the left side of the World.
5. Draw a line of south blocks across the bottom of the World.
6. Draw a line of east blocks up the right side of the World.

Your First Person World should look like this:
Exploring first person movement with bird’s eye perspective rules
Set up the Ladybug with the basic key-controlled movement rules so the 4 arrow key rules move it up, down, left, and right.

Select the Ladybug with the big arrow key, then click on the camera to switch to first person.

Use the Rotate , pan and zoom tools to adjust the camera so that it is looking over the top of the agent’s head, as if the agent were wearing a head-mounted GoPro camera. This lets the viewer see the world from the agent’s point of view.

Here is what the first person world should look like when the Ladybug is facing up and the camera has been set to look over her head:

Run the game and show the students that the basic move rules do not let the Ladybug move in a realistic way.

Ask the students to describe what the Ladybug is doing.

They should tell you that:

- The Ladybug can not turn around to reverse direction but must move backwards with no view of dangers behind it.
- The Ladybug also moves sideways like a crab rather than turning and then moving forward when the left and right arrows are typed.

Ask the students whether the Ladybug will be able to move easily through the maze in Journey or Pacman in first person.

Then, ask the students how they could make the Ladybug’s movements more realistic. The Ladybug would move in a more realistic way if it turned to face the direction it is moving.

Ask them to look at the list of basic actions to see whether there is anything useful there.

- They should notice the rotate to and rotate by actions.

These actions, and , require directions expressed as degrees on a circle.
Naming Directions in AgentCubes
AgentCubes names directions as follows:

![Diagram of a circle with degrees marked: 0°, 90°, 180°, 270°]

Directions for rotate to and rotate by actions:
- 0 degrees is North or up
- 90 degrees is West or left
- 180 degrees is South or down
- 270 degrees is East or right

Which action would make it easy to turn the Ladybug to face in the direction of movement?

*rotate to 180 0 0* turns the lady bug to face in the specified direction. Note that the number specifying direction must go in the first position.

Let the students try putting the number in either of the other positions. The agent will rotate and disappear underneath the world!

**Bird’s Eye Perspective Move Rules with Rotate To Actions Added**

Here are the 4 basic movement rules with the rotate to action added:

**Note:** the Ladybug rotates then moves because it looks more realistic!
Testing First Person with the New Bird’s Eye Move Rules
Make these changes and test the Ladybug’s movement in both bird’s eye and first person.

Tell the students that being in first person with these movement rules, although it looks as if the Ladybug should be moving straight forward, the player must keep typing the right arrow to keep the agent moving right.

Ask the students what would make movement in first person more realistic?
How do people move?
Usually we turn and then walk forward.
Ideally, the left, right and down arrows would turn the agent, while the up arrow would move the agent forward in whatever direction it faces.
Then it would be much easier to move the agent through a maze and around obstacles rapidly.

Creating First Person Move Rules
If we wish to implement this, we need a separate set of move rules for first person perspective which assign different actions to the arrow keys.

In order to switch to the First Person move rules, we will put this rule above the Bird’s Eye Perspective move rules in the Ladybug’s while running method:

![Behavior: Ladybug](image)

What would happen if this rule appeared below the Bird’s Eye Perspective move rules?
One of the regular move rules would be true whenever an arrow key was typed before AgentCubes checked whether the game was in First Person.
This rule will only become true if it is above the regular move rules!

Note: Any rules that apply in both bird’s eye and first person modes must be above this rule.

For example, win and lose rules must appear above this rule or else the game will never end if the player has an agent in first person.
**Keeping Track of the Agent’s Direction**

While playing a game in first person, the user can turn the agent many times using the left, right and down keys but when the up key is typed, the agent must move in the direction it faces. Since we will keep track of the current direction the agent is facing in an agent attribute (local variable) called Direction, we must make the Direction always stay between 0 and 359 degrees because the **rotate to** action will not work if the number of degrees is 360 or greater. Therefore, we must use the **modulo** mathematical operation (written %) to keep our Direction variable smaller than 360.

**Introduce the Modulo or Mod Math Operation to limit the value of Direction**

**Definition:** Modulo is the remainder when dividing so 5 mod 3 = 2 and 2 is the result of the modulo operation.

This remainder is always smaller than the divisor so this is a way to keep a value below a certain number.

We use mod every day when we look at a clock, which has hours marked from 1 to 12. Ask the students to tell you what time the flight will depart if you get to the airport at 10 am and have to wait 4 hours for a flight? 10 + 4 = 14 but clock time can not go past 12 so we do 14 mod 12 and know that the flight will take off at 2 pm.

In the same way, we will set the Ladybug’s Direction to the remainder of Direction divided by 360. This value, Direction mod 360 will always be a number from 0 to 359.

For example, if the agent is facing up so that Direction equals 0 and the player hits the down key twice, then types the up key, the agent will move towards the top of the AgentCubes window in the 0 degree direction because (180 + 180) % 360 = 0 degrees.

We must use the mod 360 operation because the player may type one arrow key lots of times, making the Ladybug spin in circles and adding the value of the turn to Direction over and over. AgentCubes can not rotate an agent to a direction that is 360 or larger!

Using Modulo 360 keeps the Direction Variable in the set of values \{0, ..., 359\} so that it always turns the agent to a real position.
Create the Rules for the First Person Navigation Method

The rule for the right arrow in the First Person Navigation method will look like this:

![Behavior: Ladybug](image)

The First Person Navigation method rules for the left and down arrow keys will look very similar to the rule for the left arrow key.

![Behavior: Ladybug](image)

The left, right and down arrow key rules calculate the new value of the Direction variable. What does the up arrow key rule do?

In the First Person Navigation method, the up arrow key makes the agent move in whatever direction it currently has stored in its Direction variable by calling the Move in Direction method.
Create the Move in Direction Method
Ask your students what the rules in the Ladybug’s Move in Direction Method should do?

These rules should say something like if Direction = up, move up and so on through all the directions.

Directions for rotate to and rotate by actions:
0 degrees is North or up
90 degrees is West or left
180 degrees is South or down
270 degrees is East or right

The rules must test for the numbers that stand for the different directions in AgentCubes.

Here are the rules for the Move in Direction Method.

Making Sure the Direction Agent Attribute Contains a Meaningful Value
Ask your students what will happen if the player selects the Ladybug, clicks on the First Person button, then clicks on the green play button and immediately types the up arrow.

Does the Direction attribute have a value? How does Move in Direction know which rule to call?
Actually AgentCubes sets an agent attribute to 0 if the programmer has not given it a value. In this case, if Direction = 0, then the Ladybug will move up.

What if the player had run the game for a while in bird’s eye perspective using the regular move rules and then switched to first person and typed the up arrow? Some very odd things may happen in this case such as the Ladybug moving sideways or backwards. Let the students experiment.

**Note:** It is good practice for the programmer to set the Ladybug’s Direction to a meaningful value and make sure that it always has a meaningful value.

So we should initialize the Ladybug’s Direction attribute to 0 when the Ladybug is created and we should add actions to the bird’s eye move rules to set the Ladybug’s Direction every time the Ladybug moves.

**Initializing the Ladybug’s Direction Agent Attribute**

We can set the Ladybug’s Direction agent attribute when the Ladybug is first created and before it is moved.

Create a new method and click on the word “on” in the black and yellow striped tag in the upper left corner to change it to “when creating new agent”.

![Behavior: Ladybug](image)

This method just needs a single action to set Direction to 0.

Now erase the Ladybug off your world, draw it on the world again and **save the world!** Saving means that Direction will always be set to 0 when you switch to this world.

To check the value of Direction, double click on the Ladybug and the agent attribute window will open. It should tell you the value of Direction.

**Setting the Direction in the Bird’s Eye Move Rules**

Ask the students how the regular move rules should be changed to set the Direction attribute every time the Ladybug moves.

These rules in the Ladybug’s while running method just need set actions added to them.
Here is the new move up rule from the Ladybug’s while running method:

What should the other move rules look like?

Here are the other three move rules.
Test Your First Person Move Rules
Use the big arrow tool to double click on the Ladybug agent. The Agent Attribute window should appear:

Remember what the direction numbers mean:

Turn the game on and move the Ladybug around in bird’s eye view and first person one move at a time.

Check that the value of Direction matches the way the Ladybug is facing.

If you find a mistake, go back and check the rules for misplaced arrows or incorrect direction values.

Compare your rules to the pictures above.

Adding First Person Rules to the Traveler or PacMan
If you used an alternate shape for PacMan or Frogger, just switch back to using the regular shape for your agent.

If you made a separate ladybug agent, copy and paste each of the methods you made for the Ladybug’s first person rules into the Traveler or PacMan.

- Copy and paste the rule from the Ladybug’s while running method that tests for first person and calls the First Person Navigation method into the Traveler’s or PacMan’s while running method above the regular move rules.
- Edit the Traveler’s or PacMan’s basic move rules to include the rotate to and set actions so that the agent turns to face the direction it is moving and sets the Direction to match the way it is moving.

Hooray! Now you can play Journey or PacMan in First Person!