

GameChangineer Sample Lesson Plans + Worksheets

Table of Contents

1. Getting Started
2. Setting and Object Declarations
3. Map Textbox
4. Player Character Control
5. Simple Actions, Uncond / Cond Statements
6. Border Control
7. Collision Detection
8. Pronouns
9. Bullets I
10. Scores and Winning/Losing Control
11. Power-Ups with Boolean Attributes
12. Bullets II
13. Speed and Pixels
14. Advanced Attributes and Conditionals
15. Jumping
16. Inserting Objects
17. Infinite Number of Objects
18. Timed Games
19. Wide Canvas and Side-scrolling Games
20. Advanced Topics
21. Suggested Project Ideas

Note to Teachers:

- Grade levels:
 - For grades 4 to 6: lessons 1 through 10 can be covered.
 - For grades 7 and higher: after the first 10 lessons, any subset or all of the subsequent lessons deemed appropriate can be covered for the students.
- While it is possible to rush through all 20 lessons in 5 class periods (about 5 hours), it is recommended that the teachers pace the lessons according to the students' needs to maximize learning.
- Encourage the students to write action sentences in the perspective of the characters to improve clarity and precision (see Lesson 4 for details).
- Debugging is a key skill in programming. Bugs include syntactic (spelling, grammatical) errors as well as semantic (logical, ambiguity in meaning, or unsupported terms/phrases) errors. The worksheets include debugging exercises and Game Changineer will help the user with finding and fixing errors.
- We recommend that students register using their Google accounts directly. If they do not have Google accounts, check if students can receive emails from gc.no.reply@mail – if not, ask the IT staff of your school to un-block emails in order for them to register. Alternatively, follow directions on “Need Multiple Student Accounts?” to register the students.
- Students can work in pairs or teams, especially for grades 7 and below.
- It may be beneficial to do some of the debugging exercises together as a class.
- The platform is best suited for browsers **Chrome, Firefox, Safari, and Edge**.

LEARNING OUTCOMES

Write clear instructions

Create original, exciting video games quickly

Design fun video games such as

- Space Invaders
- Pacman
- Downhill Ski
- and many more...

Compose object-oriented narratives

Reason critically about complex scenarios

Debug errors in written programs

Analyze the AI behind non-player characters

Cultivate a growth mindset

Describe the logic behind popular games

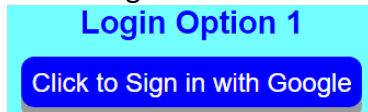
Animate scientific / mathematical concepts

Lesson 1: Getting Started

Welcome to GameChangineer! To get started, you need to register for an account from <https://gc.ece.vt.edu>

If you have a Google account (you have a Google account if you have used Google docs or used gmail to send email):

1. We recommend that users register using their Google accounts by clicking on the “Click Here to Sign in with Google” button.

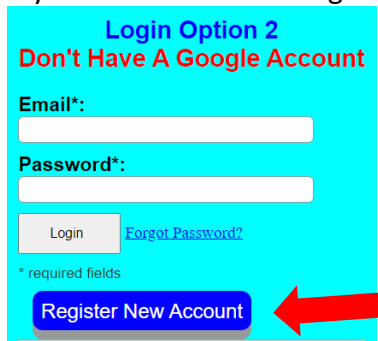


2. When you click on this button, the Google sign-in page will appear. Follow through by signing in with your Google account with your password. After successfully signing in, it will take you to the GameChangineer work page and a GameChangineer account will be created for you.
3. Skip the rest of this page and go directly to **“AFTER LOGGING IN”** on the next page.

This following section is OPTIONAL (only if you do not have a Google Account)

If you do NOT have a Google account but have another email account (such as yahoo.com, etc.) in which you can receive emails, follow the following steps:

1. If you **do not** have a Google account, click on the blue “Register New Account” button



2. Enter your name, Email address; then choose a password and enter your password 2 times (password must have **at least: 1 upper case letter, 1 lower case letter, and 1 number**)
 - a. Write down a password hint to help you remember it: _____
3. Click on the “Submit” button. An Email will be sent your Email account.

A screenshot of a "Register" form. It includes fields for "Your Full Name*", "Email Address*", "Password*", and "Retype Password*", each with a "Show" link. A "Submit" button is at the bottom. A red arrow points to the "Submit" button. A note at the top says "* required fields".

4. Open another tab in the browser and login to your Email. You should have received a new Email from gc.no.reply.mail. If you did not, please check your Junk or Spam folders.

Thanks for registering!

Your confirmation email is on its way. Please click the link in the email to complete the registration. If you do not receive the email in your Inbox, please check also your Junk (or similar) folders.

5. Open the Email and click on the **long blue link** to accept the registration. *Note: if it asks for a confirmation code, please copy and paste the confirmation code from the body of the Email.*
6. Return to gc.ece.vt.edu again and login with the account and the password that you have just set up.

If you do NOT have any Email account or you cannot receive email from gc.no.reply.mail@gmail.com:

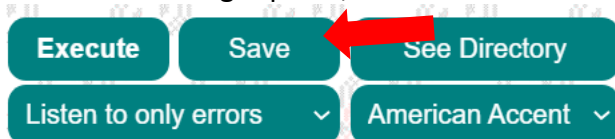
- Ask your teacher to click on the blue “Need Multiple Student Accounts?” button (see the figure above under point #1)
- Instructions will be given on the website for your teacher to register you.

Optional Section ends here

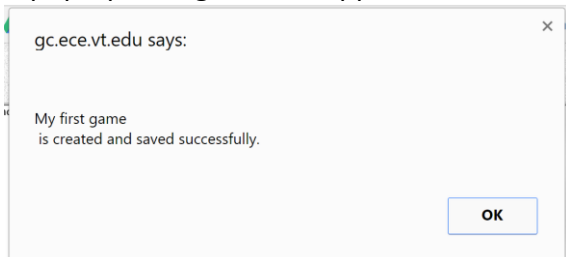
AFTER LOGGING IN:

Now that you are logged in, you are in the work page, in which you should see two large panels. The panel on the right has multiple textboxes where you can enter the text.

1. In the Title textbox: enter a title, such as **My First Game**
2. In the Your Name textbox: enter your name (if it is not already entered)
3. In the Game Plan textbox: enter **There are 5 diamonds**.
 - a. **IMPORTANT:** Make sure you do not forget a period (.) at the end of every sentence!
4. Below the Game Plan textbox, there is a button labeled “Execute”.
5. Click on the “Execute” button. You should see the contents in the left panel change. It will repeat the sentences in your game plan. If you have any typos, there will be **WARNING** or **ERROR** messages in red. If you see either of these messages, please go back to the right panel and fix your text.
6. If there are no errors, scroll down on the left panel and you should see your first game with the game’s title, your name, and the 5 diamonds on a black background.
7. Go back to the right panel, scroll down a little and click on the “Save” button



8. A pop-up dialog box will appear – click “OK”. Now your game is saved in your directory.



9. Click on the “See Directory” button to see all the games you have saved. Note: The game is saved under the title you entered for the game’s title.
10. You can click any of your saved games in the left panel, and the game will be loaded in to the textboxes in the right panel allowing you to edit or play it.

Questions:

1. Can you log in using the “Click to Sign in with Google”? _____

2. Which button in the **right panel** do you click to run / execute your game? _____
3. Are the diamonds placed at the same places every time you click 'Execute'? _____

Creating a game consists of the following 4 steps:

1. **Object declaration (setting)**
2. **Describe character control**
3. **Describe actions and interactions (plot)**
4. **Describe winning/losing conditions**

Over the next lessons, we will be covering each of these steps.

In addition to this lesson guide, there are many examples in the “Sample Video Games”. Please feel free to try them out and learn how those games are written. There are platform games (like Mario), sports games (like Target Kick, downhill ski), Pacman-style games, shoot-em up games, and more.



Finally, did you know there are **Video Lessons**? Each lesson is short, between 2 and 3 minutes. They can be very helpful for teachers and students to see the demos in action! You can find Video Lessons by clicking on the blue Video Lessons button under the Sample Video Games button. **Have fun learning!!**

Lesson 2: Setting and Object Declarations

Learning Objectives:

- Every program contains **objects** (characters); **object-oriented** style.
- Character declaration; describe how the objects are **declared** in the game.

Every video game needs to have characters or objects in it. The characters you can use in your game are shown in the right panel of Game Chingineer as the image below.



All the characters in your game are **declared** with sentences starting with “**There is**” or “**There are**”. For example, suppose you want to create a game containing **rabbits**, **carrots** and **foxes**.

- After entering the title and your name, in the idea/plan textbox, enter: **There are 3 rabbits, 10 carrots, and 2 foxes.** (Note: you need commas (,) to separate the clauses)
- Instead of the above sentence, you can also enter 3 separate sentences: **There are 3 rabbits. There are 10 carrots. There are 2 foxes.**
- Click on the “Execute” button to see what will happen on the left panel (+ “Click to Play Your Game”).

Question: Where are the characters placed in your game? _____

We can also try to place the characters in different areas.

- First, erase everything in your game plan textbox.
- Now enter: **There are 3 rabbits near the top. There are 10 carrots. There are 2 foxes near the bottom.**
- Click on the “Execute” button.

Question: Where are the characters placed in your game now? _____

Questions + Debugging Exercise:

- How would you place fifteen puppies in a game? _____
 - How would you place ten kittens **only** near the left? _____
 - What happens if you forget a period at the end of a sentence? _____
 - What would happen if you misspell a character? _____
 - Every sentence needs to have a valid character. What will happen if you write: **Today is sunny.** _____
-

Lesson 3: Map Textbox

Learning Objectives:

- Describe how the objects can be declared and placed using the **map** in the game.
- Inference and Abstraction; data representation and encoding.**

So far, the characters have been *randomly* placed on the canvas (the visible game area with the black background in the left panel). Even if you can put them in different areas with phrases like ‘near the top’ or ‘near the bottom’, it is still not very precise. What can you do if you want to place the characters in exact locations like you would in chess or Pacman? You can place individual characters more precisely using the Map Textbox (the smaller textbox below the idea/plan textbox and above the “Execute”, “Save” buttons) using the following method.

- Click on the red “**Show/Hide Map Keys**” above the Map Textbox
- An image appears that shows the single-letter keys for all the characters. For example, letter **r** represents rabbit, letter **x** represents fox, etc.

Alphabetical Character						Alphabetical Key						Example	
key	object	key	object	key	object	key	object	key	object	key	object	Map	
a	alien	@	apricot	l	ball	-	empty	a	alien	b	bird		
m	bamboo	!	bacon	b	bird	c	carrot	d	diamond	e	bone		
e	bone	k	brick	L	broccoli	f	flower	g	piglet	h	sapphire		
#	cake	c	carrot	z	cheese	i	kitten	j	elephant	k	brick		
B	cobra	Q	coyote	d	diamond	l	ball	m	bamboo	n	panda		
D	dino	+	donut	-	empty	o	rock	p	pointer	r	rabbit		
j	elephant	E	emerald	f	flower	s	spinstar	t	hamster	u	unicorn		
x	fox	N	garnet	G	gem	v	munchie	w	wall	x	fox		
t	hamster	i	kitten	A	koala	y	puppy	z	cheese	A	koala		
v	munchie	n	panda	R	pearl	B	cobra	D	dino	E	emerald		
g	piglet	p	pointer	y	puppy	F	topaz	G	gem	H	sheep		
r	rabbit	o	rock	h	sapphire	i	tiger	L	broccoli	N	garnet		
=	score	H	sheep	s	spinstar	Q	coyote	R	pearl	T	turtle		
Y	taffy	l	tiger	F	topaz	Y	taffy	l	bacon	@	apricot		
T	turtle	u	unicorn	w	wall	@	cake	+	donut	=	score		
:	clock												
						Note: lower-case l for ball upper-case i for tiger							

- You can click the “**Show/Hide Map Keys**” or the Map Keys image to close it once you are finished.
- In the map textbox, enter the following

Show/Hide Map Keys

```
C-C-C-C-C-
-l-----X
```

- Click the “Execute” button (don’t forget to enter a title for your game)

Question:

- Looking at your game, what does the letter **c** stand for? _____
- What does the dash (-) stand for? _____

The current canvas can fit at most 20 characters across and 20 characters down.

- What happens if you enter a row of characters in the map textbox that is more than 20 characters?
-

Note: the “Wall block” is a special character. It can **ONLY** be declared in the map text, but not in the game plan. The single-letter key for the wall block is **w**.

Try to draw a wall all the way around inside your game. Place some flowers in it as well. How many wall blocks are included in your game? _____

Sometimes we would like to place the objects in an orderly manner quickly, but without using the map textbox. There is a quick but imprecise way to place the objects in GameChangineer. Note that this is only for quick way of positioning – we recommend that you use the map textbox instead!

Example 1: **There are 12 diamonds aligned 3 by 4.**

Can you draw how the diamonds are positioned in the space below with “aligned 3 by 4”?

Example 2: **There are 12 diamonds spread across.**

Can you draw how the diamonds are positioned in the space below with “spread across”?

Questions:

- What is the single-letter key for the dino? _____
- What is the single-letter key for the piglet? _____
- What happens if you type **There are 3 rabbits.** (*note: rabbit is misspelled*) _____

- Are there error/warning messages for the following sentence? **There are 3 rabbits running around.** _____

- What would happen with this sentence ($4 \times 3 \neq 14$): **There are 14 diamonds aligned 4 by 3.**

-
-
- Why would you use the map textbox to place objects? _____

-
- What is the difference between 'w----w' and 'w w' in the map textbox?
-

Project: Design a jewel garden using the map textbox, containing diamonds, sapphires, etc.

Summary for Lessons 2 and 3:

- Setting involves placing characters in your game.
- Setting sentences start with "There is/are ..."
- The map textbox can be used to position characters precisely.
- If specific group of characters is not placed in the map, these characters will be placed randomly.

Lesson 4: Player Character Control

Learning Objectives:

- Describe how **player character** is declared and controlled.
- Differentiate between controlling using the **keyboard** vs. the **mouse**.
- **Cause and effect; conditional** expression.

After object declaration (Step 1), the next step is describing character control. In GameChangineer, the user can only control **one** character per game. This means that the number of instances for that character must be 1. For example: **There is one rabbit. The player controls the rabbit.**

Now you need to specify how the rabbit is controlled. Two popular sets of control keys are the arrow keys and the WASD keys. Examples are below:

- **When UP arrow is pressed, the rabbit moves up. When DOWN arrow is pressed, the rabbit... When LEFT arrow is pressed, the rabbit... When RIGHT arrow is pressed, the rabbit...**
- **When W is pressed, the rabbit moves down. When S is pressed, the rabbit... When A is pressed, the rabbit... When D is pressed, the rabbit...**

Note that all the sentences above are descriptive, just like all other sentences in Game Changineer. That is, each sentence is written in the third-person form, describing the action / interaction among the characters. This is different from *imperative* sentences, such as commands. For example, it would be wrong to write "*When the up arrow is pressed, move up.*" Since "move up" is a command, lacking a subject. Thus, remember to write each sentences as a description rather than a command.

You can also control the character with the mouse. In that case, you simply need to say,

- **The player controls the rabbit with the mouse.**

The phrase "with the mouse" indicates that the position of the mouse on the canvas determines the position of the rabbit, which means that your mouse is now the rabbit!

More importantly, the player character can be referred to as the pronoun "you". For example:

There is 1 rabbit and 2 carrots. The player controls the rabbit with the mouse.

Now consider adding the sentence: **When you touch a carrot, the carrot is eaten.** Here the pronoun "you" is interpreted as the rabbit, since you had declared that the player controls the rabbit earlier.

Think: What would happen if there were 3 rabbits declared, and the player controls the rabbits. How would the pronoun 'you' refer to?

Questions + Debugging Exercise:

- **There are 10 cheeses and 1 hamster. The player controls the hamster.** What do you need to add to the game plan so that you can control the hamster with the WASD keys? _____

- How would you make the bouncy ball game (the ball explodes when touching the bottom border) in the previous lesson like the game Pong, with a brick as the player character near the bottom? Control the brick with arrow keys. _____

- Recall from the last lesson where we discussed pronoun resolution. What would happen if the game plan is: **There is a carrot and a rabbit. The player controls the rabbit with the mouse. When the rabbit touches the carrot, you eat it.** (Question: Which object will be eaten?) _____

Summary:

- Keyboard control (Arrow keys, WASD, space bar)
- Write each sentence in the third-person form, descriptively, rather than as a command
- Mouse control
The pronoun 'you'

Lesson 5: Simple Actions, Unconditional and Conditional Sentences

Learning Objectives:

- Describe **action** and **interaction** in the perspective of the main object (**Object oriented** design).
- Distinguish **unconditional** and **conditional** sentences in cause-effect scenarios.
- Identify **antecedent** and **consequent** in a conditional sentence.
- Troubleshooting, **debugging**, and testing.

The third step of creating a game is describing actions and interactions among the characters declared in the game. Let us consider the game with **There are 3 rabbits. There are 10 carrots. There are 2 foxes.**

There are **two** ways to make characters move:

- *Unconditional*: the characters will always do this if no other conditions are met. For example, **The rabbits move right**. The rabbits will always move right, until some condition(s) that tell them (the rabbits) to do something else.
- *Conditional*: the characters will only do this if the conditional expression is met. For example, **When a fox sees a rabbit, the fox chases the rabbit**. The fox will only chase a rabbit when the rabbit is close enough to the fox. The conditional expression “**a fox sees a rabbit**” is called the **antecedent**, and the resulting action “**the fox chases the rabbit**” is called the **consequent**.

It is worth noting that the verbs “**move**” and “**chase**” must be written differently. For example, for “**move**”, we simply need a subject, such as **The rabbit moves right**. However, for “**chase**”, we need one character before and one character after the word “chase” since this movement involves two objects, such as **The fox chases the rabbit**.

- What happens if you write: **When a fox sees a rabbit, the fox chases**. _____

Similarly, for the verb “**flee from**” (or “**run away from**”), we need two characters, one before and one after the verb, since we need to tell the computer from which character is fleeing and which character to flee from.

- What happens if you write **When a rabbit sees a fox, the rabbit flees**. _____

- How could you fix the sentence? _____
- For **When the rabbit touches the carrot, it eats the carrot**. What is the antecedent? _____

What is the consequent? _____

Write in perspective of the target character: Because Game Changineer encourages you to write your game plan in an object-oriented style, it may help to for the following discussion with this neat trick! Consider the following sentence:

- When a fox sees a rabbit, the fox chases the rabbit.

We can think about it as if we put ourselves in the position of the fox. It is as if we are saying:

- *When I see a rabbit, I chase the rabbit.*

If you have trouble writing conditional statements, putting yourself in the position of the character may help! Then replace the word 'I' with the corresponding object.

Side note: In object-oriented programming (OOP) style, we focus on the objects in the program and their actions rather than inputs and outputs of the program. In other words, OOP focuses on the objects that we want to manipulate rather than the logic required to manipulate them. OOP has many benefits in data encapsulation, modular design, security, etc. In Game Changineer, each character (such as rabbit, fox, etc.) is an object. For more discussion on OOP, please refer to the Tutorials on Game Changineer website.

Sometimes you will see that the characters leave the canvas! This is because the canvas is only a very small portion of the entire space. The total space is 4 billion pixels wide by 4 billion pixels long! We will discuss how to make the characters stay inside our viewable canvas in a future lesson (Lesson 5).

Sensing distance: You might wonder, what makes an object 'see' another object. Well, there is a default distance set in Game Changineer. So if object1 is within the proximity of this default distance from another object, it can see the other character. We can change this 'sensing' distance by adding a parameter after the verb 'see'. For example:

- When a fox sees[60] a rabbit, the fox chases the rabbit.

In the above example, the fox sees the rabbit if the rabbit is within 60 pixels from the fox. In general, the default value is sufficient for most games. But you can modify this distance for your specific needs, by using the square brackets [] as shown above.

On clarity of sentences: Consider the following sentence,

The foxes chase and eat the rabbits.

While the sentence may seem clear to you, it is actually ambiguous (unclear because there are multiple possible interpretations). Do the foxes chase or eat the rabbits? If both, is there some sort of sequence (or order) of actions to be inferred?

To answer these above questions, it might help to ask yourself to write the sentence such that every verb has a distinct, separate character. How would you re-write the sentence such that every verb has its own subject? One possible solution is the following:

The foxes chase the rabbits and the foxes eat the rabbits..

Now both verbs 'chase' and 'eat' have distinct, separate subjects, and it appears that the sentence is clearer (less ambiguous). However, this sentence is still ambiguous. That is, when should the foxes eat the rabbits? Do you see this problem? Will the foxes eat the rabbits wherever the rabbits are?

If we think a bit more, a clearer way to describe the above sentence is to break down the sentence into two:

- **The fox chases the rabbit.**
- **When the fox catches the rabbit, it eats the rabbit.**

Therefore, in Game Changineer, in order to reduce ambiguity, it is required that every verb has a separate subject. When you break down sentences in such a manner, you are preparing yourself to think more like a computer scientist! Remember to **keep your sentences simple, concise, and unambiguous (clear)**.

Sharing your game: There is a “Community Showcase” button directly below the “Execute” and “Save” buttons. When your game contains at least 3 sentences and contains no bugs (no errors or warnings), then a blue “Share” button will appear to the right of the ‘Execute’ and ‘Save’ buttons (when your game canvas is shown on the left panel). If you wish to share your game, you can click on the “Share” button. If your game is NOT similar to an existing game already in the Showcase, it will be added to the Community Showcase.



Questions + Debugging Exercise:

- What does this sentence do: **The foxes wander around.** _____
- What sentence would you write to make the rabbit move up when the rabbit sees a fox?

- Describe the sentence by placing yourself as the rabbit: **When a rabbit touches a carrot, the rabbit eats the carrot.** *_When I touch a carrot, _____*
- How would you modify the above sentence to make the carrot eat the rabbit? (yes, we know it is not for real) **_When a rabbit touches a carrot, _____**
- Describe the following in Game Changineer, by putting yourself as the fox: *When I see a diamond, I move toward it.* _____
- Describe what happens if you enter the sentence: **When a rabbit sees a fox, the rabbit weeps.** _____
- Every verb needs a distinct subject to reduce ambiguity. What happens if you enter **The fox runs and chases the rabbits.** _____
- True or false? You can share a game containing the following sentence: **When a rabbit sees a fox, the rabbit weeps.** _____
- How can you fix the following sentence? **There are 10 foxes wandering around.**

- What is wrong with this sentence? **The rabbit eats the carrots.** _____

- What is wrong with this sentence? **The rocks do not move.** _____

- How can you fix the following sentence? **The foxes cannot cross the rocks.** _____

- Is the following sentence ambiguous? How should you fix it? **The foxes try to catch the rabbit.** _____

Summary:

- Unconditional and conditional sentences
- Some actions require 2 objects, such as chase and flee
- Perspective of the object
- Sensing distance

Lesson 6: Border Control

Learning Objectives:

- Describe conditional sentences on **border** events in the perspective the object.
- **Conditionals**; debugging.

This lesson discusses the interaction between characters and the borders of the canvas. In the previous lesson, the characters may 'exit' our visible canvas. To fix this, we add a conditional sentence involving the characters and borders.

Write: **There are 10 tigers.**

Next, add the following additional sentences:

The tigers wander around.

When a tiger reaches the right border, it turns around.

- What happens when a tiger reaches the right border? _____
- What happens when a tiger reaches the left border? _____

To make the tiger turn around when it reaches any border, we could add three more sentences (one for each of the left, top, and bottom borders). However, if you want to make the tiger turn around on any border, you can simply write **When a tiger reaches a border, it turns around.**

Because no specific border was indicated in above sentence, the tiger will reverse when it reaches any border. In addition to turning around, the character can do other things when reaching a border, such as:

- **When a tiger reaches a border, it blows up.**
- **When a tiger reaches a border, it stops.**
- Etc.

Now, we know that an unconditional sentence makes the character always do what is told in the sentence. However, one type of unconditional sentences only makes the character act at the beginning of the game, as shown below.

For example, **The tiger starts out moving to the right.**

This sentence is different from "**The tiger moves right.**" because of the phrase "starts out". With the phrase "starts out", the tiger will only apply this action at the beginning of the game.

Consider the following game plan:

There is one ball.

The ball starts out moving in a random direction.

When the ball reaches the bottom border, it blows up.

When the ball reaches the right border, it bounces.

When the ball reaches the top border, it reverses direction.

When the ball reaches the left border, it bounces.

What does the above game do? _____

Questions + Debugging Exercise:

- Can you make a game in which the ball bounces on all borders? If so, how would you plan your game?

- **There is a tiger.**
The tiger moves right.
When a tiger reaches a border, it turns around.
What would happen to this game plan?

- How can you fix the above game plan? (Hint: “starts out”) _____

- What is wrong with the following sentence? **When a ball reaches a border, stop.** _____

How would you fix it? _____

- What is wrong with the following sentence? **When a fox sees a rabbit, it moves to chase the rabbit.**

How would you fix it? _____

- What is the difference between the following two sentences? (Assume that you control the ball with the mouse with the sentence **You control the ball with the mouse** in the game plan already.)
 - **The tiger chases the ball.**
 - **The tiger starts out chasing the ball.**

Summary:

- Reaching any border or specific border
- Unconditional statement only at the start of the game (with “start out”)
- Computational thinking and fixing semantic errors

Lesson 7: Collision Detection

Learning Objectives:

- Describe **collision** among objects in the game, in the perspective of the main object.
- **Algorithm; conditionals; abstraction; decomposition / modularity**; debugging & testing.

Border detection in the previous lesson is a special case of collision, as we were detecting the collision between characters and borders. General collisions happen between two characters. The two characters could be of the same or different types. Verbs for describing collisions include “collide”, “touch”, “run into”, etc. Consider the following example involving two characters, the fox and the rabbit:

There is a fox and a rabbit.

The fox and the rabbit start out moving in a random direction.

When a fox reaches a border, it turns around.

When a rabbit reaches a border, it bounces.

Consider adding the sentence: **When a fox touches a rabbit, the rabbit dies.**

Here, the verb “**touch**” involves 2 characters, one before and one after it. It is the same for other verbs which describe collisions that we mentioned above. The sentence “**When a fox touches the rabbit dies**” is *incorrect* because either the word “**touches**” is missing a character, or the word “**dies**” is missing its corresponding character. Therefore, the sentence template describing collisions should look like this: **When A touches B, C dies (or does something else).**

Think: Do B and C have to be the same type? Try **When the rabbit touches a carrot, the fox dies.** _____

Another way to describe such sentences is to think in terms of ‘*what should happen*’. For example, when the rabbit touches a carrot, what should happen to the rabbit and/or what should happen to the carrot. For the carrot, we can write: **When the rabbit touches a carrot, the carrot disappears.**

When describing your game plan, you do not need to describe things that **cannot** happen. Instead, you only need to describe what needs to happen. For example, you might want to say that the tiger cannot touch a cobra. Well, this means that something must happen when the tiger touches a cobra as a punishment, right? So we simply need to say **When a tiger touches a cobra, the tiger dies / (or explodes) / ...**

Sometimes we wish to describe collisions between objects of the same type. For example, two pearls bumping into each other. There are two easy ways to describe this:

- **When a pearl collides with another pearl, they both bounce.**
- **When two pearls collide, they both bounce.**

Questions + Debugging Exercise:

- For the same rabbit and fox example above, consider adding **When the fox gets the rabbit, the game is over.** What is wrong with the preceding sentence? (Hint: ‘gets’) Will the game actually be over when the fox gets the rabbit? _____
-

- How would you describe in a game that the rabbit cannot touch a fox, or the rabbit will die? _____

-
-
- How would you write a game in which there are 3 balls? Each starts out in a random direction. The balls bounce on reaching any border. In addition, if two balls collide, they both bounce. (Hint: start with the game of the bouncing ball in the previous lesson, and add more balls to the game) _____
-
-
-
-
-

Lesson 8: Pronouns

Learning Objectives:

- Discuss how to use **pronouns** to refer to specific objects in sentences.
- Inference / analysis; debugging.

We have already used pronouns, such as “it”, in our games before. When a sentence only has one character, the resolving of the pronoun is easy. For example, “**When a ball reaches a border, it bounces.**” Here, the pronoun “it” is clearly referring to the character “ball”.

However, when there are multiple characters in a sentence, resolving pronouns is not always that simple.

Consider the following four sentences:

- **When the fox catches the rabbit, the fox eats the rabbit.**
- **When the fox catches the rabbit, it eats the rabbit.** “It” refers to _____
- **When the fox catches the rabbit, the fox eats it.** “It” refers to _____
- **When the fox catches the rabbit, it eats it.** The two “It”s refer to _____

GameChangineer will return a warning message when there are multiple pronouns in one sentence. It will ask you to clarify at least one of the pronouns to the actual character. That means you should have only one pronoun, like “it”, “they”, “you”, in each sentence to avoid confusion.

How would you re-write **When the fox catches the rabbit, it eats it.** _____

When the first character is a pronoun, it refers to the subject in the previous sentence. For example, **The fox wanders around. When it sees a rabbit, ...** Here, the pronoun “it” is bound to the subject of the previous sentence, which is the fox.

Question: What is the following pronoun “it” referring to? **The fox chases the rabbit. When it reaches a border, ...** _____

Sometimes it is simply not possible to infer the pronoun. For example, **When a fox sees a rabbit and the piglet is frozen, it ...**

Here, even a human cannot determine to what it refers. In GameChangineer, the resolution of the pronoun “it” will default to the subject of the sentence. In this case, it refers to the fox. Therefore, in general, the pronoun will refer to the subject of the sentence, or the subject of the previous sentence whenever unclear.

Questions + Debugging Exercise:

- **When a kitten sees a puppy, it flees from the puppy.** “It” refers to _____
- **When two balls collide, they both bounce.** “They” refers to _____
- **The tiger chases the cobra. When it reaches a border, ...** “It” refers to _____
- Is the following sentence correct? **When a fox sees a carrot, it chases it.** If not, how would you correct it? _____

Summary:

- To ensure clarity, only one pronoun should be used per sentence.
- Game Changineer shows how the pronoun is interpreted. If it is incorrectly interpreted, the user can change the pronoun to the exact character.

Learn from Sample Video Games on the GameChangineer website



Try the following two game sets:

- Beginner Games Set 1: Rabbit and Carrots

Question: There is a sentence, “**When the rabbit catches a carrot, the carrot is eaten.**” What would happen if we change the consequent to “it is eaten”? _____

- Beginner Games Set 2: Pong and Obstacles

Question: Consider the sentence, “**The ball starts out moving in a random direction.**” What would happen if you remove the words “starts out”? _____

Sample Video Games

Try the following sample games and see the game plan in action!

- **Beginner**

- [Beginner Games Set 1: Rabbit and Carrots](#)
- [Beginner Games Set 2: Pong and Obstacles](#)
- [Beginner Games Set 3: Dodge and Aim](#)
- [Beginner Games Set 4: Pattern Follower](#)
- [Beginner Games Set 5: Star-Dodging Bird](#)
- [Beginner Games Set 6: Multiple Shots](#)
- [Beginner Games Set 7: Good Fox, Bad Rabbits](#)
- [Beginner Games Set 8: Magic Repeller](#)
- [Beginner Games Set 9: Reaching for the Star](#)
- [Beginner Games Set 10: Maze Runner](#)

- **Intermediate**

- [Intermediate Games Set 1: Elevator Rabbit](#)
- [Intermediate Games Set 2: How Many Birds Can You Get?](#)
- [Intermediate Games Set 3: Shooting Star](#)
- [Intermediate Games Set 4: Bird Hunting](#)
- [Intermediate Games Set 5: Smart Goalie](#)
- [Intermediate Games Set 6: Extreme Dodging](#)
- [Intermediate Games Set 7: Extreme Breakout](#)
- [Intermediate Games Set 8: Save the Kittens](#)
- [Intermediate Games Set 9: Shoot and Fly in the Same Direction](#)
- [Intermediate Games Set 10: Sensing with different distances](#)

- **Intermediate + Advanced**

Lesson 9: Bullets I

Learning Objectives:

- Describe how to make a character shoot.
- Differentiate the shooting events between player-character and non-player characters.
- Describe collision event with a bullet in the **perspective** of the object.
- **Algorithm**; abstraction; decomposition; debugging.

Any character can shoot bullets. This is for both the controlled character as well as non-player characters. For the controlled character, you can shoot in addition to simply moving around. If you wish to shoot out something other than the bullet, it needs to be done by inserting a new object, which will be discussed in a later lesson (Lesson 18).

Characters cannot shoot unconditionally. Something must happen in order to trigger the shooting action. Therefore, when describing a shooting action, the sentence must be written in a conditional statement. For example:

- **When spacebar is pressed, the pointer shoots up.**
- **When the piglet sees an alien, it shoots the alien.**

Importantly, the controlled character can **only** shoot in a specific direction (up, down, left or right). That is, the controlled character cannot shoot at another character directly. This makes the games more challenging, as you would have to move and position your character at an appropriate position to aim at the target. Furthermore, any keyboard key can be used to shoot, but the mouse **cannot** be used to shoot.

The non-player characters (NPCs) can also shoot bullets, either in a specific direction, or at another character. For example:

- **When an alien is directly above the pointer, the alien shoots down.**
- **When an alien senses the pointer, the alien shoots at the pointer.**

Nevertheless, the NPCs' shooting speed is much slower, in order to make the game easy enough to play. To describe the situation when a character has been hit by a bullet, there are two ways:

- **When the alien is shot, it ...**
- **When the alien collides with a bullet, it ...**

Both of the above will do the same thing. However, the first sentence is preferred. Note that when a character has been shot, you cannot make another character die or do anything. Thus, all characters in the sentence, which describes something is being shot, must be the same. For example, you cannot say 'When a fox is shot, the dino dies.'

On clarity of sentences: Consider the following sentence

When the pointer shoots an alien, the alien blows up.

While this above sentence might seem clear to you, it is actually ambiguous. Why so? Because when the pointer shoots, the bullet may or may not hit the target. In other words, we can think of the sentence as saying:

When the pointer fires a bullet at the alien, the alien blows up.

Hence, it is essentially implying that whenever the pointer is shooting an alien, the alien will explode, irrespective of how far the alien is from the pointer or whether the bullets touches the alien. Do you get it? So the proper way is to say

When an alien is shot, the alien blows up.

Finally, the controlling player can always be shot by an NPC. However, certain NPC characters cannot be shot. These include non-predator animals puppies, kittens, unicorns, piglets, etc.

Think: What if you are controlling a puppy, or a kitten? Can you be shot by an NPC in this case? Compare the following examples to see!

Example 1:

There is a kitten and a fox.

The player controls the kitten with the mouse.

When the fox sees the kitten, it shoots the kitten.

When the kitten is shot, it dies.

Example 2:

There is a kitten and a fox.

The player controls the fox with the mouse.

When the fox sees the kitten, it shoots the kitten.

When the kitten is shot, it dies.

Can you tell what would happen differently? _____

Questions + Debugging Exercise:

Are the following sentences allowed in GameChangineer? Why, or why not?

- When a diamond is shot, the pearl blows up. _____
- When a piglet is shot, it dies. _____
- When k is pressed, the kitten shoots right. _____
- When the mouse is clicked, the alien shoots down. _____
- When the alien sees the cheese, it shoots the cheese. _____
- The alien explodes when shot. _____
- When the space bar is pressed, the rabbit shoots the fox. _____
- The alien shoots up. _____

Learn from Sample Video Games on the GameChangineer website

Try the following two game sets:

- Beginner Games Set 3: Dodge and Aim

Question: Do the following two sentences mean the same thing?

- **When two aliens collide, they both reverse direction.**
- **When an alien collides with another alien, it reverses direction.**

Explain why, or why not: _____

- Beginner Games Set 4: Pattern Follower

Question: How can you re-write the following sentence in 2 different ways using a pronoun “it”?

- **When a ball sees the alien, the ball chases the alien.**

Summary:

- The characters cannot shoot unconditionally.
- Cannot use the mouse click to shoot. You must use a keyboard keys to shoot.
- Describe collision with bullets as “When [object] is shot, ...”.

Lesson 10: Score and Winning/Losing Control

Learning Objectives:

- How to keep **score** (think of 'score' as another object in the game)
- Describe game **win/lose**.
- **Algorithm**; sequencing; **variables**; debugging / testing.

Adding a score to your game is easy. Simply describe how the score increases, or decreases, by any event. The following gives a couple of examples:

- **When an alien is shot, the score increases.**
- **When a rabbit eats a carrot, the score subtracts 2 points.**

One way to think of modifying the score is to treat "score" as a character. Therefore, the corresponding verbs "increment", "decrement", "add", etc., should come after the word "score".

You can choose where the score is displayed on the canvas by using the map textbox (discussed in Lesson 3). Use "=" to denote the position of the score. Otherwise, the score will be displayed in the upper right corner. The score is initialized to 0 by default.

You might also want to describe the conditions for winning or losing the game. For example:

- **When the score is at least 20 points, you win.**
- **When the pointer is shot, the game is over.**

Using score to count objects: Suppose you want to write the game in which that there are carrots and emeralds. The game ends when the rabbit has collected all carrots and emeralds. You can do so by the following:

There is a rabbit, 20 carrots, and 20 emeralds.

The player controls the rabbit with the mouse.

When the rabbit touches a carrot, the carrot disappears and the score adds 1 point.

When the rabbit touches an emerald, the emerald disappears and the score adds 1 point.

When the score is 40, the game ends.

Questions + Debugging Exercise:

- What would happen to the following (remember that 'score' is an object)? **When an alien is shot, you get 2 points.** _____
- How should you fix the above sentence? _____
- What would happen to the following? **When there is no more alien, the game is over.** _____

- How should you fix the above sentence? _____

Learn from Sample Video Games on the GameChangineer website

- Beginner Games Set 8: Magic Repeller

Question: How can you modify the game so that it becomes a “Magic Attractor”? _____

- Intermediate Games Set 2: How Many Birds Can You Get

Question: How can you penalize the score when a spinstar is shot?

Lesson 11: Power Ups with Boolean Attributes

Learning Objectives:

- Describe **color** and **adjectives** as Boolean attributes associated with objects.
- **Critical thinking** and logical **reasoning** with Boolean attributes to describe cause-effect relations.
- Discuss how the Boolean attributes can be used to help **transition** objects from one state to another.
- **Algorithm**; decomposition; abstraction; debugging / testing.

The characters can be in many different states, such as different colors, adjectives, etc. These states are the attributes associated with the characters. We call them Boolean attributes, because they can be either true or false only. For example, the condition “**rabbit is yellow**” means that the clause ‘the rabbit is yellow’ is true. Consider the following examples:

- **When a fox touches a rock, it becomes frozen.**
- **When the rabbit touches a gem, it is empowered. When a fox sees an empowered rabbit, it flees from the rabbit.**

In the first sentence above, the attribute “**frozen**” will be given to the fox whenever the fox collides with a rock. Note, “**frozen**” means that the character cannot move. In the second example above, the attribute “**empowered**” is given to the rabbit whenever the rabbit touches a gem.

Note that you can use any word to describe your attribute, as long as it is **not** one of those already known keywords, such as “chase”, “pearl”, etc. Thus, if the second example above were re-written as the following, it would still mean the same thing.

When the rabbit touches a gem, it becomes xyz. When a fox sees a xyz rabbit, it flees from the rabbit.

In addition, any attribute must appear in at least two sentences. Specifically, it appears in the consequent of one conditional sentence, and in the antecedent of another conditional sentence.

Question:

Can you spot the two occurrences of “xyz” in the above example? _____

The name of the attribute can also be composed of two words, joined by an underscore (_). For example,

When an empowered rabbit touches a sapphire, it becomes more_empowered.

Here, the attributes empowered and more_empowered are two different attributes.

Now, let us re-consider the very first example in this lesson again. You might wonder how a fox can become unfrozen. There are two ways:

- **When a fox touches a rock, it freezes. Otherwise, the fox is not frozen.**
- **When a fox touches a rock, it freezes for 3 seconds.**

In the first method, the keyword “**Otherwise**” indicates the opposite of the conditional expression in the previous sentence. In this example, it means the condition that when the fox does not touch a rock. Therefore, whenever the fox is not touching a rock, it is not frozen.

In the second method, we attach an amount of time to the attribute. Therefore, in this example, the fox will be “frozen” for 3 seconds. After 3 seconds, the attribute “frozen” will automatically become false.

Now we can write a more elaborate game with a rabbit, carrots, and foxes, in which the rabbit can be empowered. Try the following Example game plan and see what would happen:

There is a rabbit, a fox and a diamond.

The player controls the rabbit with the mouse.

When the rabbit touches a diamond, the diamond disappears and the rabbit is empowered for 4 seconds.
When the fox sees the rabbit that is not empowered, it chases the rabbit.
When the fox sees an empowered rabbit, it flees the rabbit.

Note that when attributes are included in a complex conditional expression, such as “**When the fox sees an empowered rabbit**”, it would be difficult to determine the opposite of the antecedent if the next sentence starts with “**Otherwise**”. The “**Otherwise**” could mean any of the following:

- **When the fox does not see an empowered rabbit, ...**
- **When the fox sees a rabbit that is not empowered, ...**
- **When the fox does not see any rabbit, ...**

Because the negation of such an antecedent is ambiguous/unclear, it is recommended to avoid using “**Otherwise**” when such complex conditional expressions are used. Instead, write out the complete conditional expressions for the other cases. In this example, choose any of the three instead of ‘otherwise’.

Questions + Debugging Exercise:

- What is wrong with the following sentence? **When the rabbit touches a wall, it does not become red.**

- Will the rabbit ever die in the following game plan? Try it out and see.

There is one rabbit and 10 spinstars.

The rabbit wanders around.

When the rabbit touches a spinstar, the spinstar disappears and the rabbit is empowered for 4 seconds.

When the rabbit touches a spinstar and the rabbit is empowered, the rabbit dies.

Why, or why not?

- Write a complete game plan using the Example game plan of a rabbit, a fox and diamonds, which the rabbit becomes empowered when touching a diamond (middle of the previous page).

-
-
-
- How would you describe a game plan involving 2 rabbits, 10 carrots, and 1 fox, so that the fox will only chase the rabbit when the rabbit has eaten the carrot? (Hint: first attach an attribute to the rabbit when the rabbit has eaten the carrot. Then, describe the action of the fox about chasing the rabbit with the attribute.) A possible solution is shown below:

There are 2 rabbits, 10 carrots, and one fox.

The rabbits wander around.

When a rabbit touches a carrot, the carrot is eaten.

When a rabbit touches a carrot, the rabbit becomes empowered.

When a rabbit is empowered, the fox chases it.

Learn from Sample Video Games on the GameChangineer website

Try the following two game sets:

- Intermediate Games Set 3: Shooting Star

Question: Do the following 2 sentences mean the same thing?

- **When a white alien is shot, the alien explodes.**
- **When an alien that is white is shot, the alien explodes.**

Explain why, or why not: _____

- Intermediate Games Set 4: Bird Hunting

Question: Explain why the following two cases mean the same thing. _____

- **When the rabbit collides with spinstar, the rabbit eats the spinstar and the rabbit is empowered for 5 seconds.**
- **When the rabbit collides with spinstar, the rabbit eats the spinstar. When the rabbit collides with spinstar, the rabbit is empowered for 5 seconds.**

Summary:

- Every Boolean attribute must appear in at least 2 sentences
- The name of an attribute must be unique
- The name of an attribute can be formed by joining multiple words with underscore (_), such as abc_xyz
- The name of an attribute can contain numbers, such as empowered3.
- Boolean attributes can be used for a limited amount of time
- Computational thinking and logical reasoning with Boolean attributes
- Use of “otherwise”
- Modifiers with Boolean attributes, such as “the empowered rabbit” or “the rabbit that is not empowered”

Lesson 12: Bullets II

Learning Objectives:

- **Combine** Boolean attributes with bullet events.
- **Critical thinking** and reasoning.
- **Algorithm; debugging / testing.**

Now that we know about Boolean attributes (lesson 11) and bullets I (lesson 9), we can design more complicated shooting games. For example, how would you make the alien explode after it has been shot twice? GameChangineer purposely does not allow you to write, **“When an alien is shot 2 times, it blows up.”** because it wants you to exercise computational thinking by adding attributes.

Therefore, one way to think about this is to attach an attribute to any alien that has been shot one time. Then when it is shot again, it will blow up. Let us use color attributes in the example for now so that we can visualize the effects as well. For example,

When an alien is shot, it turns green.

When a green alien is shot, it blows up.

What do you think about the above sentences? *When an alien is shot, it will turn green. Next, if the alien happens to be green when it is shot, it will blow up.* Because the game proceeds frame by frame, the instant that an alien is shot, it will turn green in the next frame. However, in the next frame, the bullet is already gone. Therefore, in order to get the alien to explode, we must wait for a second bullet. Do you see it? Try writing a game plan and experiment with it! You can start with the above alien example. _____

Now, try to make the alien blow up only after it has been shot 3 times. Would the following example work?

There is an alien and a pointer.

The user controls the pointer using the mouse.

When the pointer sees the alien, the pointer shoots up.

When the alien is shot, it turns green.

When a green alien is shot, it turns yellow.

When a yellow alien is shot, it blows up.

Your answer: _____

What should we do if we want the alien blow up after 2 seconds? Again, we can use attributes to help us here. For example,

When an alien is shot, it becomes green.

When an alien is shot, it also becomes has_been_shot for 2 seconds.

When a green alien is not has_been_shot, it blows up.

In this example, we use 2 attributes, “green” and “has_been_shot” (note that every attribute has to be a *single word without* spaces). When an alien is shot, it takes on both attributes. However, one of the two attributes, “has_been_shot”, only lasts for 2 seconds. Then, after 2 seconds, only the “green” attribute remains. The third sentence shown above will blow up the alien since it is “green” but is no longer “has_been_shot”. Do you get it? Try writing a complete game plan and experiment with it! You can start with the above example. _____

Questions + Debugging Exercise:

- Are the following 2 sentences saying the same thing? _____
 - **When a green alien is shot, it blows up.**
 - **When an alien that is green is shot, it blows up.**What does the pronoun 'it' refer to? _____
- What is the difference between the following two sentences? _____

 - **When a green alien is shot, it blows up.**
 - **When an alien that is not green is shot, it blows up.**
- Are the following 2 sentences saying the same thing? _____
 - **When a green alien is not has_been_shot, it blows up.**
 - **When an alien that is not has_been_shot is green, it blows up.**

Challenge: Can you design a game in which the alien blows up 2 seconds after it has been shot for the second time? _____

Challenge 2: Can you design a game in which the alien blows up when it has been shot 2 times within 1 second? _____

Summary:

- Use of Boolean attributes to describe complex shooting scenarios
- Computational thinking and reasoning with Boolean attributes

Lesson 13: Speed and Pixels

Learning Objectives:

- Describe different **speed** for different objects.
- Editing and modifying **positions**.
- Algorithm; **variables**; abstraction; debugging / testing.

So far, the speeds of the characters have never been changed in a game. However, they can be modified. You can specify the speed of a character in two different ways:

- **The speed of the fox is 2 pixels per frame.**
- **The fox chases the rabbit at 2 pixels per frame.**

The first sentence is a declaration, which initializes the speed of the fox. The second sentence sets the speed of the fox for the chase action. Note that the unit for speed is pixel(s) per frame. In a game, the frame rate is 50 frames per second, meaning that the frame refreshes 50 times in one second. The canvas is 600 pixels wide. At 2 pixels per frame, the fox would be moving 100 pixels every second initially. This means that it takes 6 seconds for the fox to travel across the canvas. Consider the following scenario:

There is a fox and a rabbit.

When the fox sees the rabbit, it chases the rabbit at 2 pixels per frame.

Otherwise, it wanders around at 1 pixel per frame.

When a fox meets any border, it bounces.

What can you observe about the speed of the fox?

What does the following sentence do? **The fox moves at 2 pixels per frame.** _____

Note that above sentence can also be written as “**The foxes wander around at 2 pixels per frame**”!

What is wrong with this sentence? **The fox moves at 2 pixels per second.** _____

What do the following sentences do?

- **When the fox reaches a rock, it moves down 20 pixels.**

- **When the fox reaches a rock, it moves down 20 pixels per frame.**

Without “**per frame**”, it simply takes one stride! However, you might want it in this way sometimes. You can re-position the character in any direction.

How would you move aliens down 25 pixels when they reach a border? _____

Learn from Sample Video Games on the GameChangineer website

- Intermediate Games Set 6: Extreme Dodging

Question: How would you end the game when a carrot is shot? _____

- Intermediate Games Set 7: Extreme Breakout

Question: Is it necessary to declare that there are 54 rocks in the game plan (if they are already placed in the map)? _____

Lesson 14: Advanced Conditionals and Attributes

Learning Objectives:

- Describe **complex conditionals** involving more than two antecedents.
- Introduction of **variable attributes**.
- Practice **critical thinking** and reasoning.
- Algorithm; **variables**; **abstraction**; decomposition; debugging / testing.

There are many different ways to write the same idea. For example, the following three sentences are all saying the same thing:

- **When a green alien is shot, it explodes.**
- **When an alien that is green is shot, it explodes.**
- **When an alien is green and the alien is shot, it explodes.**

However, the first two sentences are preferred because they are regarded as a single condition, while the third sentence has a compound condition.

To avoid run-on sentences, only 2 antecedents are **allowed** in a conditional expression. You may ask, what if you want to include 3 cases in a conditional expression? This is easy to do. You can add Boolean attributes as modifiers. For example:

- **When a fox sees an empowered rabbit, the fox becomes scared.**
- **When a scared fox collides with a rock, it explodes.**

In the above example, we say that the fox will explode if 3 conditions are true, namely:

- the fox sees a rabbit
- the rabbit is empowered
- the fox collides with a rock

To handle these 3 conditions, we add an adjective "**scared**" to describe the fox when it sees an empowered rabbit. Then, we combine the "**scared**" attributes with the third condition to complete the game plan.

Note that compound conditions joined by "**or**", such as "**When the fox collides with a rock or the fox collides with a brick, ...**", should be broken into two separate sentences:

- **When the fox collides with a rock, ...**
- **When the fox collides with a brick, ...**

So far we have learned to use score to keep count of the number of carrots eaten. What if we would like to change the color of the rabbit when the rabbit has eaten a certain number of carrots? In such cases, score is insufficient, because when score is used as an antecedent in a conditional sentence, it can only be used for determining game win or lose.

Thus, in this lesson, we will cover a new topic: **variable attributes**. Like Boolean attributes, variable attributes are associated with each individual object. However, its values are not just true or false, but take on any value, such as 3, 5, or 17. If their values are strictly integers, sometimes they are also referred to as 'integer attributes.'

To describe a variable attribute, we need to say "attribute of the object". For example, "the num_carrots_eaten of the rabbit".

Suppose we are interested in writing a game in which the rabbit collects carrots on the field. When 10 carrots are collected, the rabbit turns yellow. When 15 carrots are collected, the rabbit turns pink. How would you describe this game? Here is an example of using variable attributes:

There is a rabbit and 20 carrots.

The player controls the rabbit with the mouse.

When the rabbit touches a carrot, the carrot disappears and the num_carrots_eaten of the rabbit adds 1.

When the num_carrots_eaten of the rabbit equals 10, the rabbit turns yellow.

When the num_carrots_eaten of the rabbit equals 15, the rabbit turns pink.

When all carrots are eaten, the game ends.

You can also show the values of variable attributes on the canvas. To do so, simply add "**Display num_carrots_eaten of the rabbit.**" to the game plan.

Sometimes you might want to combine variable attributes with other conditions. For example, when the num_carrots_eaten of the rabbit is greater than 10 and the rabbit sees a fox, then ... However, such compound antecedents involving variable attributes are not allowed. You must first convert the variable attribute to a modifier clause. In this running example, we would write:

When the rabbit whose num_carrots_eaten is greater than 10 sees a fox, the rabbit ...

In the above sentence, the variable attribute is converted to a modifier phrase "whose num_carrots_eaten is greater than 10". Of course, there are other ways to write this, such as converting the variable attribute to a Boolean attribute first. In this case we would write:

When the num_carrots_eaten of the rabbit is greater than 10, it becomes empowered.

When an empowered rabbit sees a fox, it ...

Questions + Debugging Exercise:

What does the following do?

There is a rabbit.

The rabbit is controlled with the mouse.

When the rabbit is visible, the position_y of the rabbit is 500.

How would you write a game plan in which a bird only dies if it has:

- (1) met a fox, (2) collided with a rock, (3) been shot
-
-

How would you write a game plan in which the rabbit can eat both apricots and carrots?

- How would you display both the number of carrots eaten and the number of apricots eaten on the canvas? How would you end the game when all of apricots and carrots are eaten? (Hint: first convert variable attributes to Boolean attributes)

Learn from Sample Video Games on the GameChangineer website

- [Beginner Games Set 9: Reaching for the Star](#)
- [Beginner Games Set 10: Maze Runner](#)
- Intermediate
 - [Intermediate Games Set 1: Elevator Rabbit](#)
 - [Intermediate Games Set 2: How Many Birds Can You Get?](#)
 - [Intermediate Games Set 3: Shooting Star](#)
 - [Intermediate Games Set 4: Bird Hunting](#)
 - [Intermediate Games Set 5: Smart Goalie](#)
 - [Intermediate Games Set 6: Extreme Dodging](#)
 - [Intermediate Games Set 7: Extreme Breakout](#)
 - [Intermediate Games Set 8: Save the Kittens](#)
 - [Intermediate Games Set 9: Shoot and Fly in the Same Direction](#)
 - [Intermediate Games Set 10: Sensing with different distance](#)
- Intermediate + Advanced
 - [More Than 50 Other Intermediate and Advanced Games](#)

- [Advanced Games: Rabbit and Foxes 2](#)

Question: How would you make the fox explode when the empowered rabbit catches it? _____

Summary:

- Clever use of Boolean attributes to describe complex conditional expressions
- Variable attributes

Lesson 15: Jumping

Learning Objectives:

- Describe games with 2.5D
- How to **land** on objects.
- **Algorithm**; decomposition; abstraction; debugging / testing.

In many games, the characters can jump. To do this in GameChangineer, we need to view the canvas as if it has a height. Such games are called 2.5D, since it is between 2D and 3D games. For example, *Super Mario* is a 2.5D game. Jumping motion is easy to do. Just use the verb “jump” when some condition is met. For example,

- **When up arrow is pressed, the rabbit jumps.**
- **When a fox collides with a ball, it jumps.**

Note that when describing the jumping motion, we do not need to indicate the direction of the target object. The direction is implicit, as we can only jump upward. We cannot direct a character to jump at any other character since that target character may be far away. Also, for characters that can jump, gravity is animated for this character. What do you think the following will do?

- **There is a rabbit. The rabbit moves east. The rabbit jumps.**

You guessed it. The rabbits will continuously jump eastward.

When you are in 2.5D, you should avoid moving the object up *without* using the jumping verb, as the gravity will likely prevent it from moving up.

Do you think the rabbit would jump in the following case? Why, or why not? _____

- **There is a rabbit. The rabbit moves up. The rabbit moves down.**

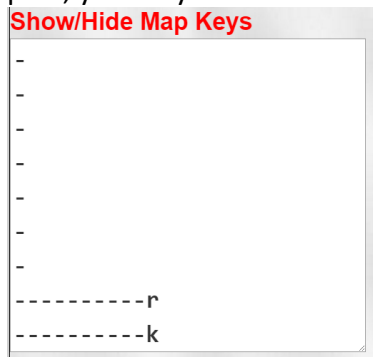
Collision detection involving jumping motion is a bit tricky. For example, suppose we want to say that a rabbit destroys a fox by stomping on it. Otherwise, regular collisions would kill the rabbit. We can say it this way:

- **When the rabbit stomps on a fox, the fox explodes.**
- **When the rabbit collides with a fox, the rabbit explodes.**

The above two sentences distinguish between two different types of collisions. Write a complete game plan and find out the differences! _____

There are certain objects, such as “**bricks**”, that will automatically stop the falling when an object lands on them. Also, the bottom border of our canvas can stop the falling too.

Example 1: Try the following example with map textbox to see how “brick” and “rabbit” works. In the game plan, you only need “**Rabbit jumps.**”



Example 2: Would the rabbit in the following example disappear from the visible canvas? Try it out. _____

- **There is a rabbit. The rabbit jumps.**

Most other objects will not stop object from falling, unless you explicitly describe it. For example,

- **When the rabbit lands on a sapphire, the rabbit stops.**

You can also change the height an object jumps. This is similar to the moving speed in a previous lesson. For example: **When space bar is pressed, the rabbit jumps at 10 pixels per frame.**

An object can jump again only after it has landed, unless if the character can fly, such as a “**bird**”. In this case, it is possible for it to jump again before it lands.

Questions:

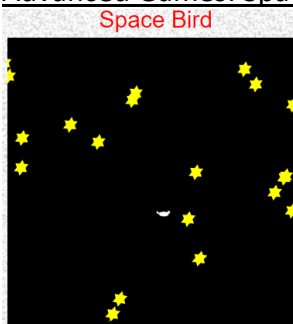
- Use the map textbox to insert several **topazes**. The player character is the **rabbit**. Write a game plan that lets the rabbit land on the topazes. _____

Learn from Sample Video Games on the GameChangineer website

- Intermediate Games Set 1: Elevator Rabbit

Question: Change the **bricks** to **sapphires**. What other sentences do you need to add to make it similar to the original game with bricks? _____

- Advanced Games: Space Bird



You may have noticed that the bird can jump even without landing on anything. This is because the bird can fly, as we have discussed earlier!

Question: Try the following two games and see the difference:

- There is a rabbit. You control the rabbit. The rabbit moves right. When spacebar is pressed, the rabbit jumps. When the rabbit reaches a border, it wraps around.
- There is a rabbit. You control the rabbit. The rabbit flies right. When spacebar is pressed, the rabbit jumps. When the rabbit reaches a border, it wraps around.

Try to press the spacebar more than once before the rabbit lands. Which game lets the rabbit jump without landing? Why? _____

Summary:

- Jumping and landing
- Fly and jumping (double jumping)
- Changing height of jump by changing the speed (“..., the rabbit jumps at 10 pixels per frame”)

Lesson 16: Inserting Objects

Learning Objectives:

- Describe insertion events based on conditional sentences.
- Different insertion between player and non-player characters.
- Algorithm; decomposition / modularity; abstraction; reasoning; debugging.

You might have wondered if a player character can insert another character on the canvas or not. The answer is yes! This can be done easily:

- **There is a panda. The player controls the panda.**
- **When spacebar is pressed, the panda inserts a bamboo.**

Where do you think the bamboo will be inserted? _____

Without moving the panda, you might not be able to see the inserted bamboo. Consider adding the sentence **“The bamboo moves randomly.”** to make the inserted bamboos emerge from behind the panda!

The non-player characters (NPCs) can also insert objects. Consider adding the following sentences to above example:

- **There is a puppy. The puppy moves randomly.**
- **When a puppy sees the panda, it inserts a bone.**
- **The bone moves toward the panda.**

What do you think the above sentences will do? _____

It is important to notice that the object making the insertion should be the same as the first character in the antecedent. For example, you CANNOT say **When the puppy is green, the kitten inserts a flower**, since the kitten is different from the puppy.

To make it clearer, try the following game plan to see what happens.

There is a puppy, a panda and a kitten.

The player controls the panda with the mouse.

The puppy and kitten move randomly.

When a puppy sees the panda, the kitten inserts a flower.

Does it work as expected? _____

Note that the rate of insertion is kept at approximately one insertion per 0.5 second to avoid cluttering the canvas.

Finally, we can also insert an object at a location different from the inserting object. To do so, we place two numbers within brackets [] after the verb ‘insert’. For example:

- **When a puppy is green, it inserts[0, 50] a bone.**
- **When the kitten is red, it inserts[1, 35] a flower.**
- **When a rabbit is happy, it inserts[4] a carrot.**

In the above, the first number within the brackets is direction, and the second number is distance from the inserting object. For direction, 0=above, 1=right, 2=down, 3=left, 4=random. So the first sentence would make the puppy insert a bone 50 pixels above the puppy. In the second sentence above, the kitten would insert a flower 35 pixels to the right of the kitten. In the third example, the rabbit will insert a carrot at a random location.

What would the following sentence do?

- **When the koala is red, it inserts[3, 65] a cheese.**
-

Questions + Debugging Exercise:

- Would the following sentence work?
When a puppy sees the panda, it inserts a bone.
If not, how would you correct it?
-
-

How do you design a game that is like space invaders, but where the invaders shoot down diamonds, instead of bullets? (Hint: some invaders will insert diamonds when a condition is met, and the diamonds move down at 3 pixels per frame) _____

Learn from Sample Video Games on the GameChangineer website

- Advanced Games: Feed the Rabbits

Question:

- What happens when you click on top of a rabbit? _____
- What happens when you click on top of a fox? _____

- Advanced Games: Lead the Puppy

Question: What is the difference between the following 2 sentences?

- **When d is pressed, the diamond inserts a bone at its location.**

- **When d is pressed, insert a bone at its location.**

Answer and explain: _____

Summary:

- Inserting object with the controlled player using the keyboard
- Inserting objects with non-player characters
- Inserting objects must not depend on a relational event

Lesson 17: Infinite Number of Objects

Learning Objectives:

- Use of **invisible** objects.
- Insertion of objects using invisible objects.
- **Algorithm**; decomposition; abstraction; debugging / testing.

Suppose you wish to write a game in which the player controlled rabbit goes out to find and collect carrots. Initially, there might be 30 carrots on the field. This simple game might look something like the following:

There is one rabbit and 30 carrots.

The player controls the rabbit.

When an arrow key is pressed, the rabbit moves in the same direction.

When the rabbit touches a carrot, the carrot is eaten.

Now, the above game plan is functionally working. Every time when the rabbit touches a carrot, the carrot disappears. Suppose you wish to replenish the carrots on the field. That is, whenever a carrot gets eaten, another one appears at a different location on the field. How could we do that?

When the rabbit touches a carrot, the rabbit inserts[4] a carrot.

By adding the above sentence, we note that the rabbit will insert a new carrot at a random location (with the [4]) when it eats a carrot. Pretty simple, right?

We can accomplish the same goal with an alternative, slightly more complicated, method. This is where computational thinking comes into play. Let us say that there exists an invisible character, which just randomly moving around on the canvas. Whenever the rabbit eats a carrot, this invisible character inserts a new carrot wherever it happens to be at that time. We can do so by using Boolean attributes. Add the following game plan to the above example:

There is a turtle.

The turtle is invisible.

The turtle starts out moving in a random direction.

When the turtle reaches a border, it reverses direction.

When the rabbit touches a carrot, the turtle becomes notified.

Otherwise, the turtle is not notified.

When the turtle is notified, it inserts a carrot.

The above addition to the game plan involves an invisible turtle. The invisible turtle wanders around the canvas and never leaves the canvas. It attains the Boolean attribute “**notified**” whenever the rabbit touches a carrot. Note that the next sentence starting with “**Otherwise**” is important. Remember why? Without this sentence, the turtle will be notified forever once the rabbit touches one carrot.

Try combining the two sections of the game plan above and see if it works as it should. You can make the turtle visible to see how it insert the new carrot when an old one got eaten by removing the sentence **The turtle is invisible**.

Now, since the turtle can only insert approximately two new carrots per second, the total number of carrots on the canvas may not remain at 30. This is because it is possible for the rabbit to eat 2 or more carrots within half a second, but the turtle will still only insert 1 new carrot during that time. In other words, the rabbit may eat carrots faster than the turtle can insert carrots.

Questions + Debugging Exercise:

- How would you insert the carrots only in the top half of the canvas? (Hint: add an invisible wall across the middle, and the turtle reverses direction when hitting the wall)

- Can you design a game that flowers bloom when a puppy finds a bone? (Hint: you can start with putting bones, a puppy and an invisible bird on the canvas. Then design a game plan that whenever the puppy touches and eats a bone, the bird inserts a flower.)

Summary:

- Use invisible objects to insert new objects to give an illusion of replenishing objects
- The rate of insertion by non-player characters is slow to avoid cluttering up the canvas.

Lesson 18: Timed Games

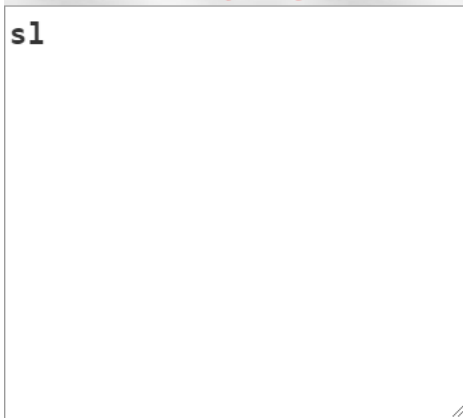
Learning Objectives:

- Use of Boolean attributes to time various events.
- Describe a visible clock using objects.
- **Algorithm**; abstraction; decomposition; debugging / testing.

What should you do if you want to limit the amount of time for a player to achieve a target goal? Instead of defining the time limit in the setting, we encourage the learner to use **computational thinking and logical reasoning** to solve this problem.

Essentially, we need a timer of some sort. Recall that we can set an attribute of an object for a specified amount of time. Perhaps we can take advantage of this for our purpose in this lecture. Using the map textbox, we place the spinstar (s) at the upper-left corner and the ball (l – “lowercase L”) directly to the right of the spinstar, as shown.

Show/Hide Map Keys



Consider the following sentences:

- **The ball moves left.**
- **When the ball collides with the spinstar, the ball blows up and the spinstar turns red for 5 seconds.**

What will happen when we run these sentences?

When the game begins, the ball immediately collides with the spinstar and explodes, and the star turns red for exactly 5 seconds.

Now, suppose we want to write a game in which the player controls a rabbit to accomplish some task within 5 seconds. We can add the following sentences:

- **When the spinstar is red, the rabbit is racing_against_time.**
- **When the rabbit is racing_against_time and the spinstar is not red, the game is over.**

The first sentence above says that whenever the spinstar is red, the rabbit will have the attribute called “racing_against_time”. Remember that the attribute has to be a single word! Therefore, we combine racing against time into a single word by using the underscore “_”. However, you can use whatever symbols, other than system key words, for your attribute names. The rabbit will have this attribute whenever the spinstar is red.

The next sentence is also key to time-control your game. What will happen to the rabbit's attribute “racing_against_time”, when the spinstar is no longer “red”? Nothing will happen, since we did not say how to set the rabbit to not “racing_against_time” when the spinstar is not “red”! In other words, the rabbit will *remain* “racing_against_time” even when the game is over. Do you get it?

Now let us look at the second bulleted sentence above. Remember the spinstar only stays “red” for 5 seconds. Thus, after 5 seconds, the spinstar is no longer “red” (turns back to the original color), but the rabbit is still “racing_against_time”. As a result, both conditions in the antecedent are true. Therefore, the game is over. What would you do if, instead of using the spinstar's color, you want to use the “ball” to time? Consider the following sentences:

- **When the ball is gone, the rabbit is racing_against_time for 5 seconds.**
- **When the rabbit is not racing_against_time, the game is over.**

The above two sentences look correct on the surface, but they are actually **logically incorrect** for timing the rabbit. Can you explain why? _____

The first sentence says that **When the ball is gone, the rabbit is racing_against_time for 5 seconds**. However, the antecedent of “ball is gone” is always true after the ball explodes. Therefore, the 5 seconds timer for the rabbit's “racing_against_time” attribute is always being reset to 5 seconds. The timer will never count down to zero. Keep this in mind!

Visible Diamond Clock

There are other ways to design a time-limited game. For example, if we place a diamond in the upper-left corner with a ball immediately next to it (on the right of the diamond).



Then, we can add the following sentences:

The ball starts out moving right.

When the ball reaches a border, it turns around.

When the ball collides with the diamond, game is over.

What do you think will happen? _____

This allows us to see the ball rolling across the screen and back, implicitly letting the player see how much time is left. We can also change the speed of the ball to increase or decrease the total time of the game.

Learn from Sample Video Games on the GameChangineer website

- Advanced Games: Timed Rabbit

Question: Can you modify the timer so that the game ends when an object reaches a border?

- Advanced Games: Speedy Turtle

Question:

- In this game, does the turtle ever move up or down? _____

- Are accelerating and decelerating names for Boolean attributes? _____

Summary:

- Visual clock with moving objects
- Timed game with attributes
- Computational thinking to reason about timed games

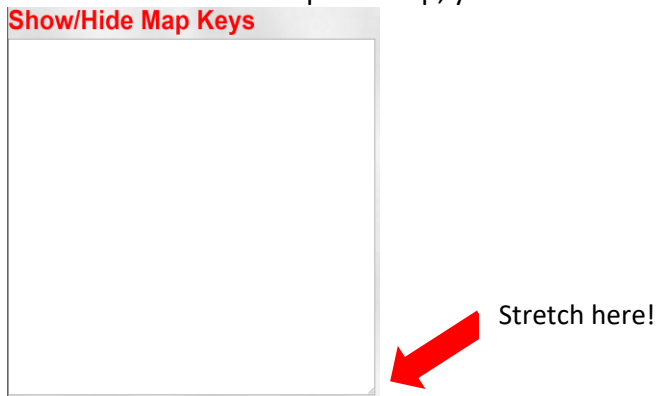
Lesson 19: Wide Canvas and Side-Scrolling Games

Learning Objectives:

- Describe side-scrolling platformer games with wide screen.
- Abstraction; decomposition; analysis; reasoning.

Suppose you would like to make a game in which the player character constantly moves right through the maze consisting of spinstars. How would you design it?

You can first create the maze using the map textbox. Then you can increase the map size by creating a larger maze. To view the complete map, you can stretch the lower-right corner of the map textbox.



Let the player character be an alien. Because we want to move the character constantly to the right, we can add an unconditional statement "**The alien moves right.**" The user only controls the up and down directions:

- **When the up arrow is pressed, the alien moves up.**
- **When the down arrow is pressed, the alien moves down.**

In addition, the spinstar would make the alien explode.

- **Whenever the alien touches a spinstar, it blows up.**

Thus, the complete game plan is as follow (without using the map):

Wide canvas with 5 frames.

There are 80 spinstars scattered.

There is one alien. The player controls the alien.

The alien moves right.

When the up arrow is pressed, the alien moves up.

When the down arrow is pressed, the alien moves down.

When the alien touches a spinstar, it blows up.

When the alien is gone, the game is over.

Try it to see if it works!

Did you notice that you do not see 80 spinstars on the canvas at first? It is because they are scattered all over the wide canvas of 5 frames. Therefore, on the average, there will be about $80/5 = 16$ spinstars on each viewable game screen.

Now, can we write a game similar to this **without** using a wide canvas? Yes!

Instead of moving the alien always to the right, make the spinstars always move to the left. When any spinstar reaches the left border, it wraps around with new coordinates to give an illusion that you are moving to the right! In addition, since we just describe a single frame, instead of 5 frames, we just need one-fifth of the spinstars (16 spinstars, instead of 80!).

The alternative game plan looks like the following:

There are 16 spinstars scattered.

There is one alien.

The player controls the alien.

The spinstars moves left.

When a spinstar reaches the left border, it wraps around with new y coordinates.

When the up arrow is pressed, the alien moves up.

When the down arrow is pressed, the alien moves down.

When the alien touches a spinstar, it blows up.

When the alien is gone, the game is over.

Do you notice the difference between the above two game plans?

Can you design a side-scrolling game where the player character can jump onto different platforms?

Learn from Sample Video Games on the GameChangineer website

- Beginner Games Set 10: Maze Runner (2nd one in the set)

Question: Can you make the fox die after being shot 2 times? _____

- Advanced Games: Mario Rabbit

Question: Instead of shooting down bullets by the birds, can you make the birds drop diamonds

instead? (Hint: think inserting an object) _____

Summary:

- Side scrolling game with wide canvas
- Design the map with wider rows
- The screen automatically scrolls when the controlled character is within 100 pixels of the border.

Lesson 20: Final Advanced Topics (briefly discussed)

Learning Objectives:

- Describe one object turning into another object
- Describe conditional sentences probabilistically
- Use of advanced **variable** attributes
- **Abstraction**; decomposition; algorithm; debugging / testing.

Topic 1: Object Conversion

Sometimes we would like to convert an object to another type of object. For example, when a rabbit touches a carrot, the carrot turns into a diamond. In GameChangineer, object conversion can only be done via the state of the object, not by a relational event (like touch or collide). Therefore, for the example of turning a carrot into a diamond, we need to say:

When the rabbit touches a carrot, the carrot becomes mutated.

When a carrot is mutated, it turns into a diamond.

Topic 2: Probabilistic Actions

So far, all actions are carried out as long as the antecedent is satisfied (true). However, sometimes we might want to perform an action probabilistically. For example, consider the case where the fox chases the rabbit probabilistically when it sees the rabbit. The game plan would look like:

There are 2 rabbits and 4 foxes.

The rabbits start out moves right.

When a rabbit reaches a border, it turns around.

The foxes wander around.

When a fox sees a rabbit, it chases the rabbit with 5 percent probability.

What do you think will happen?

When you run the above game plan, you will see that the foxes chase the rabbits more than 5 percent of the time. Why is that? Recall that we are describing the game plan for every frame of the video game. Therefore, the game is evaluating the game plan 50 times a second. Therefore, whenever a fox sees a rabbit, it has 50 chances to decide whether it wants to chase the rabbit or not every second! So the actual probability would be $1 - [(100\% - 5\%)^{50}]$, which is about 92.3%. This is **much** higher than the 5% used in the game plan! Keep the frame refresh rate in mind when you are designing action with probability.

Note that the word “**percent**” is necessary; it cannot be replaced with the “%” symbol.

Topic 3: Advanced variable attributes

So far, the entire game plan is independent of user-defined variables. You might have wondered, since the plot in the game plan only describes the actions for any time instant, how would one go about describing different behaviors for a specific time-instant?

For instance, how would you describe a rabbit moving in a shape of a square in the game plan? That is, it moves to the right first, then down, then left, and finally up.

Since the action of the rabbit is for any time instance, we cannot tell it to go right, left, up and down all at once. We must have something to indicate which way it is supposed to move.

Although we have used *Boolean attributes* to denote an internal state of an object, it is difficult to use them to track how far the rabbit has moved in any given time instant. One way to fix this is to use attributes that are independent of any frame.

To use such attributes, we need to say the "attribute of object". Let us take "**the state of the rabbit.**" as an example. The sentence "**The state of the rabbit is initially 0.**" will set the attribute of the rabbit to 0 in the beginning (in the setting). In the plot of the game plan, we can use the attribute "state" (and other additional attributes, if any) to help to describe how characters act and interact. So, while the Boolean attribute can only be True or False at one time, the attribute of object can be integer numbers 0, 1, 2, etc.

To move the rabbit in the shape of a square, consider the following:

There are two rabbits.

The state of the rabbit is initially 0.

The num_steps of the rabbit is initially 0.

When the state of the rabbit is equal to 0, it moves right.

When the state of the rabbit is equal to 1, it moves down.

When the state of the rabbit is equal to 2, it moves left.

When the state of the rabbit is equal to 3, it moves up.

The num_steps of the rabbit increases every frame.

When the num_steps of the rabbit is greater than or equal to 50, the state of the rabbit increments and the num_steps of the rabbit becomes 0.

When the state of the rabbit is greater than 3, the state of the rabbit becomes 0.

What do you think the above will do?

Note that we are using these new attributes of "state" and "num_steps" to help us define moving in a square. The above description is at a low-level of abstraction compared to simply saying, "**The rabbit moves in a square of size 50 pixels.**" Note that at a lower-level of abstraction, the simple action of 'moving in a square of size ...' needs to be described using multiple sentences! This is the reason that games written in GameChangineer contain far fewer sentences than the actual program code.

For any actions that GameChangineer does not support, you can generally revert to describing your intent using a lower level of abstraction! In so doing, you also get to practice problem solving by breaking the problem down to smaller steps.

We can use variable attributes for many other things as well. For example, suppose we want to design a game where a rabbit eats carrots. But you win when half of the carrots are eaten. We can do so in the following:

There are 20 carrots and one rabbit.

You control the rabbit.

When an arrow key is pressed, the rabbit moves in the same direction.

When the rabbit touches a carrot, the carrot is eaten.

When a carrot is eaten, the num_carrots_eaten of the rabbit increases.

When the num_carrots_eaten of the rabbit is 10, you win.

Finally, there are also built-in variable attributes, such as "position_x", "position_y" that hold the current horizontal and vertical position coordinates of the character. In addition, "size" holds the size of the character.

Question: What do the following game plans do?

There is a rabbit.

The time_left of the rabbit is initially 50.

The time_left of the rabbit decreases every frame.

When the time_left of the rabbit equals 0, game is over. _____

There is a fox and a rabbit.

When the position_x of the rabbit is greater than the position_x of the fox, the rabbit moves left.

When the position_x of the rabbit is less than the position_x of the fox, the rabbit moves right.

Question: How can you display the number of objects in the top half of the canvas using variable attributes?

(Check out the Tutorials link on “Computational Thinking – How many rabbits are in the top half” to see how it can be done!) _____

Question: How would you design a game where the objects go faster and faster? (Check out the Tutorials

link on “Computational Thinking – Faster and faster” to see how that can be done!) _____

In addition to this lesson guide, there are more examples in the “[Sample Video Games](#)” on the GameChangineer website that were not fully discussed. Please feel free to try them out and learn from how the games are written. There are platformer games (like Mario), sports games (like Target Kick, downhill ski), Pacman-style games, and shoot-em ups, etc.

Topic 4: Subsets and indices

Sometimes we want to have only a subset of one type of characters that does something. For instance, suppose there are 10 rabbits. If we say “Some rabbits move right”, it is unclear which of the 10 are included in the “some”. Remember that in programming, we must not be ambiguous when describing what we wish to do.

We can specify which instances we want to include in GameChangineer. In many programming languages, the counting of instances starts with 0. Therefore, if there are 10 instances of rabbits in total, the first instance of the rabbits is rabbit #0, and the last one is rabbit #9. To specify a single instance in the set, simply use square brackets immediately after the character, such as rabbit[2] -- this is the rabbit instance #2. On the other hand, to specify a range, we need two numbers within the brackets, such as rabbits[2,6] -- this includes rabbit instance #2, #3, #4, #5, and #6.

Consider the following game plan:

There are 10 rabbits.

Rabbit[0] moves down.

Rabbits[1,3] move right.

Rabbits[4,9] move left.

What do you think would happen?

- Rabbit #0 will _____.
- Rabbit instances #1, #2, and #3, will _____.
- Finally, rabbit instances #4 to #9, will _____.

In addition, when used in antecedents of conditional statements, [5, 10] means anyone in the range of instances between 5 and 10. If there are pronouns, the instance index from the antecedent will be carried over to the pronoun. For example, consider the following:

There are 10 rabbits and 5 foxes.

The foxes move randomly.

When rabbit[0] is not dead, it turns yellow.

When rabbits[2,5] see a fox, they flee from the fox.

Here, the antecedent "**When rabbit[0] is not dead**" is about the single instance rabbit #0. The other statement, "**When rabbits[2,5] see a fox, they flee from the fox.**", means that when **any** rabbit in instances #2 through #5 sees a fox, it will flee from the fox.

- What would rabbit instance #5 do (since it is included in 3 sentences)?

There are 10 rabbits.

When rabbit[5] is not dead, it becomes yellow.

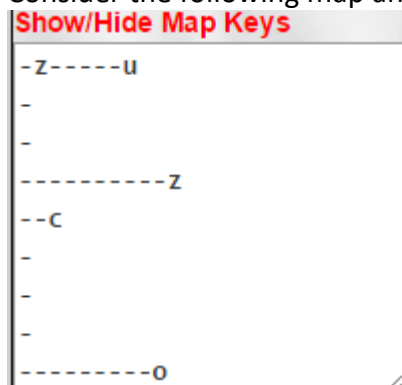
Rabbits[0,5] move right.

Rabbits[5,9] move left.

Topic 5: Guide Posts

We can do many things with Boolean attributes (Lesson 11). We have seen how attributes help to make the target blow up after 2 shots, as well as make the foxes flee when the rabbit is empowered. In this lesson, we will see how attributes can help us to direct the path of a character.

Consider the following map and the following game plan:



The unicorn starts out moving to the left.

When the unicorn touches a cheese, it becomes happy. Otherwise, it is not happy.

When the unicorn is happy, it moves down.

When the unicorn encounters a carrot, it becomes happier. Otherwise, it is not happier.

When the unicorn is happier, it moves to the right.

When the unicorn touches a rock, it becomes happiest. Otherwise, it is not happiest.

When the unicorn is happiest, it moves left.

Try it and see what the unicorn does! _____

We need the “**Otherwise**” statements because we need to remove the attributes whenever they are not needed anymore. If not, the unicorn will be “happy”, “happier” and “happiest” at the same time when it reaches the rock (in our case, the unicorn may not reach the rock though. Try it!). In that case, it would not be clear which direction the unicorn should move.

Only remove the sentence “**Otherwise, it is not happy.**” What do you observe? _____

Only remove the sentence “**Otherwise, it is not happier.**” What do you observe? _____

Only remove the sentence “**Otherwise, it is not happiest.**” What do you observe? _____

Remove all three “**Otherwise**” sentences. What do you observe? _____

Compare the four modified examples. Can you explain what was happening in each case and why?

Now, there is a built-in attribute called “invisible”. We can make some or all of the guide posts invisible, and the unicorn will still follow the same route! Simply adding the following sentences into your game plan,

The cheeses are invisible.

The carrot is invisible.

The rock is invisible.

What happens after you add these 3 sentences? _____

Questions + Debugging Exercise:

- Can you design a game where the character always moves in a square (Note: you need to place guide posts in the map textbox)? _____

Have fun learning!!

Lesson 21: Sample Project ideas

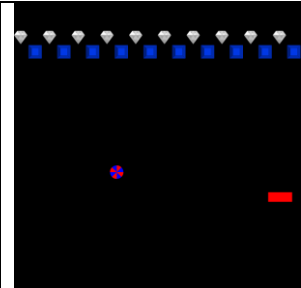
While there are many mini games included in the first 22 lessons, and there are many examples from the Sample Video Games link that you can use, below are additional suggested projects that can be used to enhance learning:

Project 1: Break-out with ball and paddle (on completion of lesson 8)

- **Simple break-out**
(less than 10 sentences)
- **Break-out with different target objects and/or multiple balls**
(about 15 sentences)

Grading Rubric:

1. Setting clarity: 30%
2. Plot clarity and creativity: 50%
3. Control character clarity: 20%

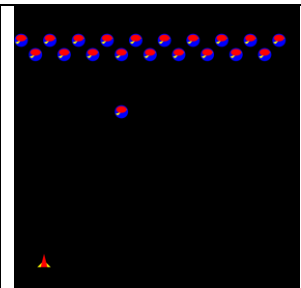


Project 2: Shoot'em up games (on completion of lesson 10)

- **Basic Space Invaders**
(less than 10 sentences)
- **Invaders that attack by coming down toward the player**
(less than 15 sentences)

Grading Rubric:

1. Setting clarity: 30%
2. Plot clarity and creativity: 50%
3. Control character clarity: 20%
4. Optional/Extra credit: Boolean attributes of invaders



Project 3: Pacman-like games with map (on completion of lesson 11)

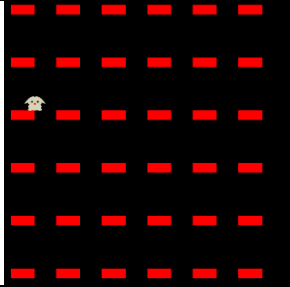
- **Simple maze game**
(less than 15 sentences)
- **Pacman with power jewels**
(15 to 25 sentences)

Grading Rubric:

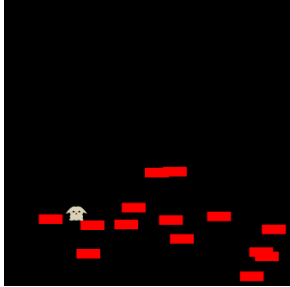
1. Setting clarity and maze construction/aesthetics: 30%
2. Plot clarity, creativity, clear differentiation of chase / flee / wander: 50%
3. Control character clarity: 20%
4. Optional/Extra credit: Boolean attributes of predators / controlled character



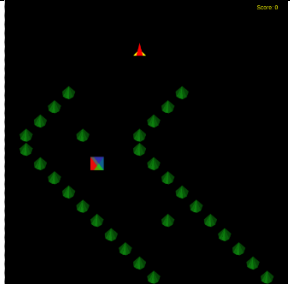
Project 4: Jumping character games (on completion of lesson 17)

<ul style="list-style-type: none"> • Simple jumping onto various platforms (10 to 15 sentences) • Jumping with different gems to collect (15 to 20 sentences) <p>Grading Rubric:</p> <ol style="list-style-type: none"> 1. Setting clarity and layout aesthetics: 20% 2. Plot clarity, creativity, landable objects, etc: 30% 3. Control character clarity, jumping control: 20% 4. Clever use of Boolean attributes: 30% 	
--	---

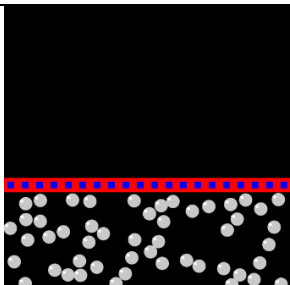
Project 5: Wide-canvas platformer (on completion of lesson 21)

<ul style="list-style-type: none"> • Simple platformer (less than 15 sentences) • Platformer with different collectibles and enemies to avoid (20 to 30 sentences) <p>Grading Rubric:</p> <ol style="list-style-type: none"> 1. Setting clarity and maze construction/aesthetics: 20% 2. Plot clarity, creativity of rewards, etc.: 60% 3. Control character clarity, movement, jumping, etc: 20% 4. Optional/Extra credit: Boolean attributes of objects 	
--	---

Project 6: Sports games (on completion of lesson 22)

<ul style="list-style-type: none"> • Downhill ski (less than 15 sentences) • Target kick (20 to 30 sentences) • Kick ball (20 to 30 sentences) <p>Grading Rubric:</p> <ol style="list-style-type: none"> 1. Setting clarity and layout aesthetics: 20% 2. Plot clarity, creativity of target/goal, etc.: 30% 3. Use of Boolean and Variable attributes: 30% 4. Control character clarity, movement, jumping, etc: 20% 	
--	---

Project 7: Science animations (on completion of lesson 22)

<ul style="list-style-type: none"> • Gravity / physics animations (can vary in size) • Biological illustrations (can vary in size) <p>Grading Rubric:</p> <ol style="list-style-type: none"> 1. Setting clarity and creativity: 30% 2. Plot clarity, creativity of movement, target, etc.: 40% 3. Use of Boolean and variable attributes: 30% 	
---	---