

Game Engines

09

from A to Z



How to Use
This Program

Software is changing the world. The programs installed in computers and apps that make it more convenient to use smartphones are all software. Software is in every part of our lives, so it is difficult to find areas where we are not affected by software.

The state-of-the-art science and technology that we see in the news is also helped by software. In turn, progress in math, science and technology advances software further. As such, math, science and technology are closely related with and cannot be separated from software.

These module series were created through collaboration between experts in related fields and software education, and its suitability for classrooms has been verified. As students follow teachers' direction through each module, they will be able to better understand the world that has been changed by software.



CONTENTS

Intro	Game Engines from A to Z	02
PART1	Traditional Games Revisited	08
Activity1	Building a Board Game Based on Yut Nori	12
PART2	How Game Characters Move	16
Activity2	Aircraft Collision Detection	24
Activity3	Making a State Diagram	26
PART3	Becoming a Game Developer	28
Activity4	Building a Game by Using a Game Engine	30
Activity5	Making Flappy Bird by Using GameSalad	36
Career Exploration	Game Developer	40
Maker Activity	A robot catapult and a catapult game engine	42



Game Engines from A to Z

Whether with a computer or a smartphone, many of us are enjoying games almost everyday. There are various kinds of games, from simple puzzle games to complex strategy games with rich visual elements. How are these games made? The answer is by using "game engines."

Then, what is a game engine? Is it similar to an automobile engine? From here on out, we will look further into game engines.

WHAT IS A GAME ENGINE?



When you play games, you see a variety of scenes on a screen: characters collide with each other; bombs are dropped and explode in your enemy's camp, and so forth. Some games are composed of only quadrangular backgrounds and characters. A game is a sophisticated "computer program" that determines what happens within a variety of possibilities.

Making a program is called "programming" or "coding," and the content of a program which consists of various conditions and commands is called a "source code." Programming requires knowledge of programming languages.

Game engines: a quick and easy tool for making a game

Just knowing a programming language is not enough. In order to create various game scenes, you need scientific knowledge

such as how two objects move when they collide with each other and what kind of changes happen when an object drops from high above. You also need to know how to depict such scenes realistically, as well as how gunshots, bomb explosion, wind, etc. sound.

Based on such various forms of knowledge and many possible situations, you make a game (program) that controls your computer's sound card, graphic card, and display devices. This is far from easy.

Thus, as a result of efforts to build games more easily, a number of "game engines" have been created.

Let's assume you build a large and complex house by using Lego blocks. Instead of placing the blocks one by one, it would be easier to build a house if someone has already made bigger units such as a living room



and a bathroom. You simply have to assemble the units as desired. A game engine is just like a livingroom or bathroom module. A slight modification to a game engine can result in a new game. No programming is needed regarding the motions/sounds or graphic details of objects.

Some game companies develop unique game engines and sell them or sometimes provide them for free. These are called “commercial engines.” Used by many game developers, commercial engines are known to have relatively few errors. Companies can make games more quickly by using commercial game engines than by developing their own ones.

The use of game engines

Large game companies with sufficient financial resources develop and use their own engines that can provide the best output for their planned games. However, smaller companies with less money tend to purchase commercial engines.

Lineage II, one of Korea’s representative games, was built using the Unreal Engine 2. (Source: NCSOFT)

Then, is a game more likely to fail if a commercial engine has been used rather than its company’s own one? Not necessarily. Although a game engine is essential in making a game, the success of a game often depends on a creative scenario and/or game rules. Admittedly, companies with their own engines could also afford to have their games developed by uniquely talented game developers and promoted/advertised

widely. Thus, they have higher chance of producing successful games.

Choosing a game engine

Game companies are not the only ones that can make games. If you have brilliant game ideas and are able to use a game engine to some degree, you can also make a game by using a commercial engine and you can even sell that game.

There are a lot of game engines, but the most representative ones are: Unreal Engine, Unity, CryEngine, and Cocos2d. Those four engines are all commercial engines.

Unreal Engine

The Unreal Engine (UE) is a game engine developed by Epic Games in 1994, first showcased in the 1998 game Unreal. One of Korea’s representative games Lineage II uses UE2 released in 2001. This second version was also used in games such as Ragnarok 2, Priston Tale 2, and Blade & Soul. Released in 2004, UE3 is an extended version made to adapt to various game genres. In March 2015, a free version of UE3 was released, following a free version of UE4 in 2014. Although these versions are basically free to use, you are required to pay Epic Games 5% of possible gross revenue resulting from the use of one of them if the revenue exceeds a certain limit.



THE 4 BEST GAME ENGINES

Unity

Released in 2005 by Unity Technologies, Unity was initially a dedicated game engine for Apple's Mac computers. Since 2010, however, Unity has gained popularity as it came to be used for smartphones as well as video game consoles such as Xbox 360 and Wii.

One of the greatest advantages of Unity is that you can easily apply games to multiple platforms such as Windows, macOS, Android, and iOS. Previously, in order to convert a game, for example, an Android game into an iOS game, you had to develop a program that supports iOS, which required a lot of time and money. However, the Unity engine just requires you to buy a program called a "plugin" for a desired platform and to simply execute it. In addition, while other game engines cost tens of thousands of



dollars, Unity only cost about 1,000-2,000 dollars, making it an ideal engine for small to mid-sized video game companies and individual developers.

Moreover, Unity comes loaded with various game characters and backgrounds for free or with charges so that developers with less skills in the area of creating characters/backgrounds or programming can easily make a game. Since the release of Unity 4 in 2012, the number of Unity game developers around the world has reached about two million and games that use the Unity engine have rapidly increased as well. With the release of Unity 5 in 2015, Unity is provided free of charge for personal, non-commercial uses.

Just like the case for the Unreal Engine, however, game companies have to pay Unity Technologies a certain part of their possible gross revenue as royalty. Popular games that use the Unity engine include Gamevil's Kritika (a mobile game) and Activision Blizzard's Hearthstone.

Cocos2d

Cocos2d is a game engine developed in Argentina in 2008 by a group of developers.

Since then, the game engine has been diversified into many variants. Cocos2d-x, developed by Chinese programmer Zhe Wang, is one such variant that allows a single source code to be easily applied to various platforms. Due to its versatility which enables even beginners to make 2D games easily,



MapleStory M, a mobile game developed by Nexon, was also made with the Unity engine. © Nexon

Cocos2d-x is loved by individual game developers.

Popular games that use Cocos2d-x include Cookie Run and Everybody's Marble. The game engine is provided free of charge.



Cookie Run was developed by using Cocos2d-x. The game engine is free of charge.

CryEngine

CryEngine is a game engine developed by the German game company Crytek. In March 2004, CryEngine 1 was released along with the game Far Cry, followed by CryEngines 2 and 3. In 2013, CryEngine without a version number was released. Games that were developed by using CryEngine include NCSoft's Aion: The Tower of Eternity, Wemade's Icarus, and Nexon's Warface.



CRYENGINE®

Making a game is getting easier

So far, we have taken a brief tour of the four most popular game engines.

When game engines were first introduced, developers had to write program codes one by one for configuration settings. They also had to type in commands in order to check the results of game execution.

Today's game engines, however, are very easy to use. A developer can move a game character by simply dragging the character with a mouse and can immediately check the result of code changes. Installing a game engine is also straightforward: just download one from the Internet and execute it.

As developing a game has become easier, you can realize your ideas in the form of a game without difficulty by using a game engine that is suitable for your ability and the characteristics of the planned game.

Unfortunately, no representative game engines have been developed in Korea to date. If you are interested in game engines, how about trying to develop a innovative game engine that will contribute to the growth of Korea's game industry and lead the global market?

PART 1

Recommended target
Elementary creative experience activities

Relevant subjects
Practical course

Traditional Games Revisited

How long did it take for the number of telephone users to reach one billion since its invention? About 75 years. What about the case of Candy Crush Saga? Only 15 months! With the development of IT, games have become an important part of entertainment, enjoyed by almost everyone. By the way, do you know any traditional fun games that have been played a long time before?



The number of players of Candy Crush Saga (mobile and desktop versions) reached 1 billion in just 15 months.

Yut Nori, a traditional Korean board game

Although not as popular as before, Yut Nori (meaning “the game of yut”) is played with family members and/or neighbors during traditional holidays. The game is known to have been played as early as the Three Kingdoms era.

The game’s rules are not very complicated. Each team casts four yut sticks at the same time and counts the sticks that are turned over (flat side up). One stick turned over and three sticks facing up (round side up) is called “do” (1 point). Two sticks turned over is called “gae” (2 points). Three sticks turned over is called “geol” (3 points). All sticks turned over is called “yut” (4 points), while all sticks facing up is called “mo” (5 points). Each score corresponds to the number of stations that a game piece called “mal” can advance. The board consists of 29 stations, and there are usually four game pieces for each team. The team who brings all their pieces home first wins the game. If a piece lands on a station occupied by a piece from their own team, the two pieces can travel together from that point on. If a piece lands on a station occupied by their opponent’s team, the opponent’s piece is



Yut Nori is a traditional Korean board game played by casting four yut sticks and moving game pieces called “mal.”
© National Folk Museum of Korea

removed from the course.

Now, let's summarize the game rules. The rules must not be contradictory and must address all possible situations.

①
②
③
④
⑤
⑥
⑦

“Do,” “gae,” “geol,” “yut,” and “mo” represent the pace of pigs, dogs, sheep, cows, and horses, respectively. This is why “do” results in a single space advancement, while “mo” results in 5 space advancement. Generally played in the first month of the year, the game is also known to have been used in fortune-telling.

What do you think about the idea of making and servicing an on-line version of a Korean traditional game such as Yut Nori? What are the pros and cons?



“Do,” “gae,” “geol,” “yut,” and “mo” represent the pace of pigs, dogs, sheep, cows, and horses, respectively.

Pros	
Cons	



The ancient Egyptian game Senet is one of the oldest known board games in human history.

Senet, a board game from ancient Egypt

Games have long been enjoyed in the West as well. Dating back to c. 3500 BC, the ancient Egyptian game Senet is one of the oldest known board games in human history. The game is played with four sticks (resembling Korean yut sticks) or a dice, a game board with a grid of 30 squares, and two sets of game pieces (five of each). Usually two people play the game.

The game's rules are simple. You throw four sticks or a dice in the air. The front and back side of the sticks have been marked beforehand. The number of sticks that land with front side facing up determines the number of squares you can move at a time. Four back sides facing up means six moves. There is no rule for five moves because five was regarded as an unfortunate number by the ancient Egyptians.

Whoever gets all the pieces off the board wins the game. You can place only one piece per square. If a piece lands on a square occupied by the opponent, the two pieces switch their places. However, you cannot do this if two or more of the opponent's pieces are next to each other. Besides, your piece cannot move beyond three consecutive pieces of the opponent.

Now, let's summarize the game rules. The rules must not be contradictory and must address all the possible situations.

①

②

③

④

⑤

⑥

⑦

How should the game pieces be placed and moved in order to win the game? Write down your own strategy.

e.g.) Make sure that two or more of my pieces are next to each other.



Senet game boards and pieces have been found in many tombs of Pharaoh. Ancient Egyptians believed that they must win a game with a demon that guards a gate to the afterlife in order to pass through it.
© Wikipedia

Senet game boards and pieces have been found in many tombs of Pharaoh. It is said that school teachers in ancient Egypt asked their students to play Senet as a homework. From royal family members to slaves, every ancient Egyptian played Senet. To them, Senet meant more than a fun game. They believed that they must win a game with a demon that guards a gate to the afterlife in order to pass through it.

In terms of game elements and rules, Korean Yut Nori and Egyptian Senet have similarities and differences. Write them down in the table below.

+ Play Senet at:

<http://www.mindgames.com/game/Senet>



Similarities		
Differences	Yut Nori	
	Senet	

Activity ①

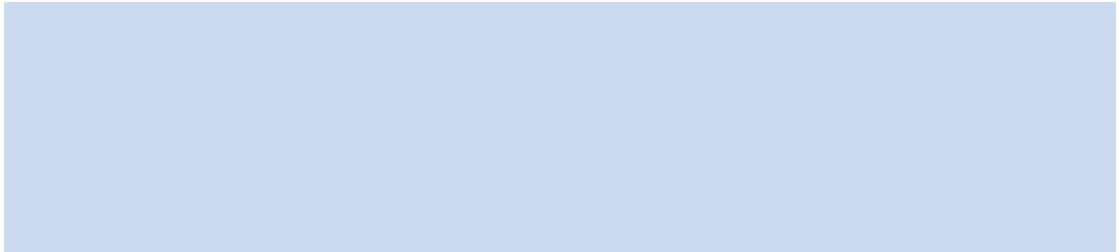
Building a Board Game Based on Yut Nori

Yut Nori consists of a game board, yut sticks, and game pieces. You can build your own board game by modifying some of these elements and applying special missions (such as rewards or penalties) to certain stations.

What to prepare Materials 1-①~1-③, black/colored pencils, eraser, scissors, glue

Activity

- 1 Summarize the rules of your new board game as you did for Yut Nori and Senet.



- 2 Draw a game board on [Material 1-①]. The board can resemble a Yut Nori board, but it is recommended that you make your own unique board. Apply special missions (such as rewards or penalties) to certain stations.

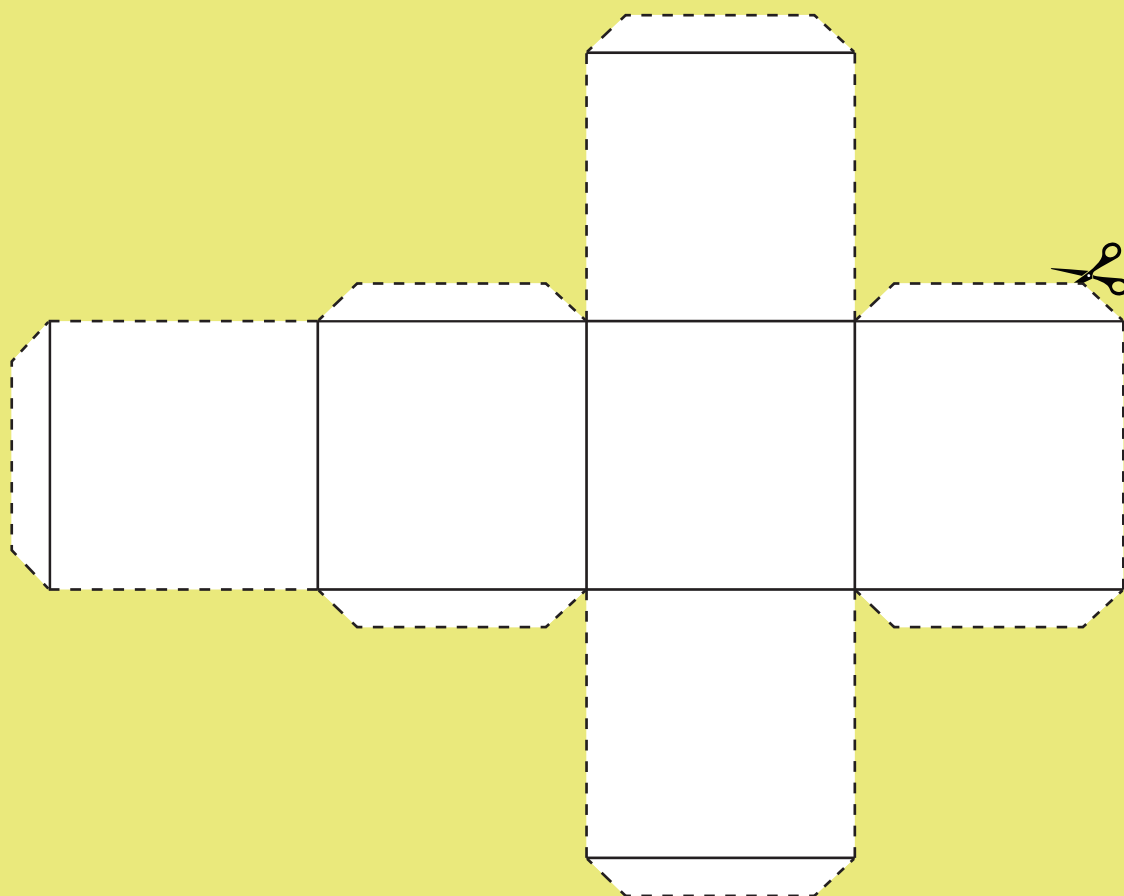
- 3 Cut out the figure on [Material 1-②] to make a cube, which is the equivalent of yut sticks. Mark the number on each side of the cube.

- 4 [Cut out the figures on [Material 1-③] for game pieces. Decide how many will be used.

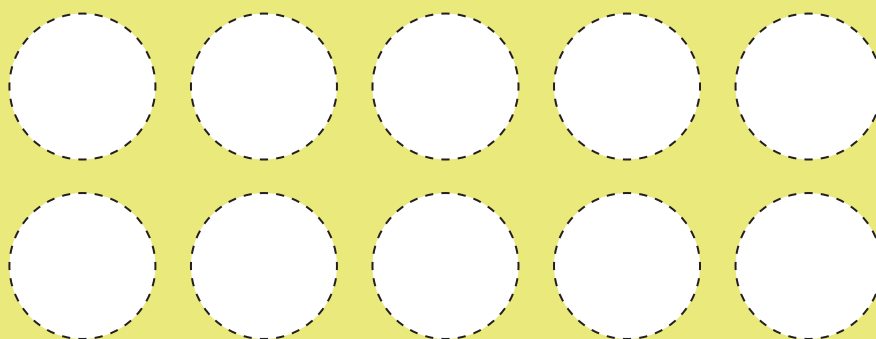
- 5 When making a board game, it is more effective to decide who does what (such as the game designer, rule master, game tester, etc.). See the table below and identify your role.

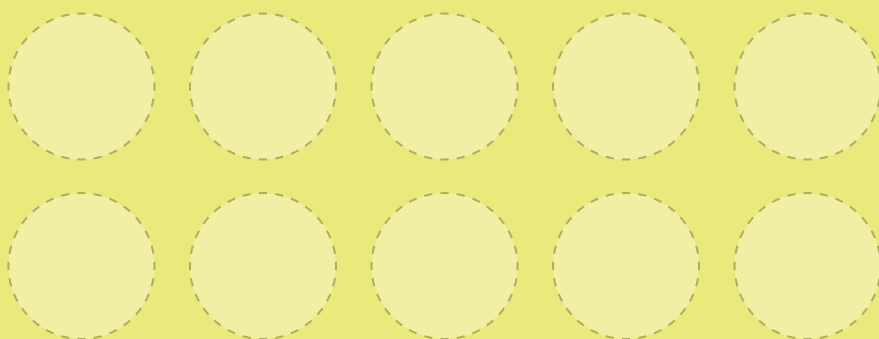
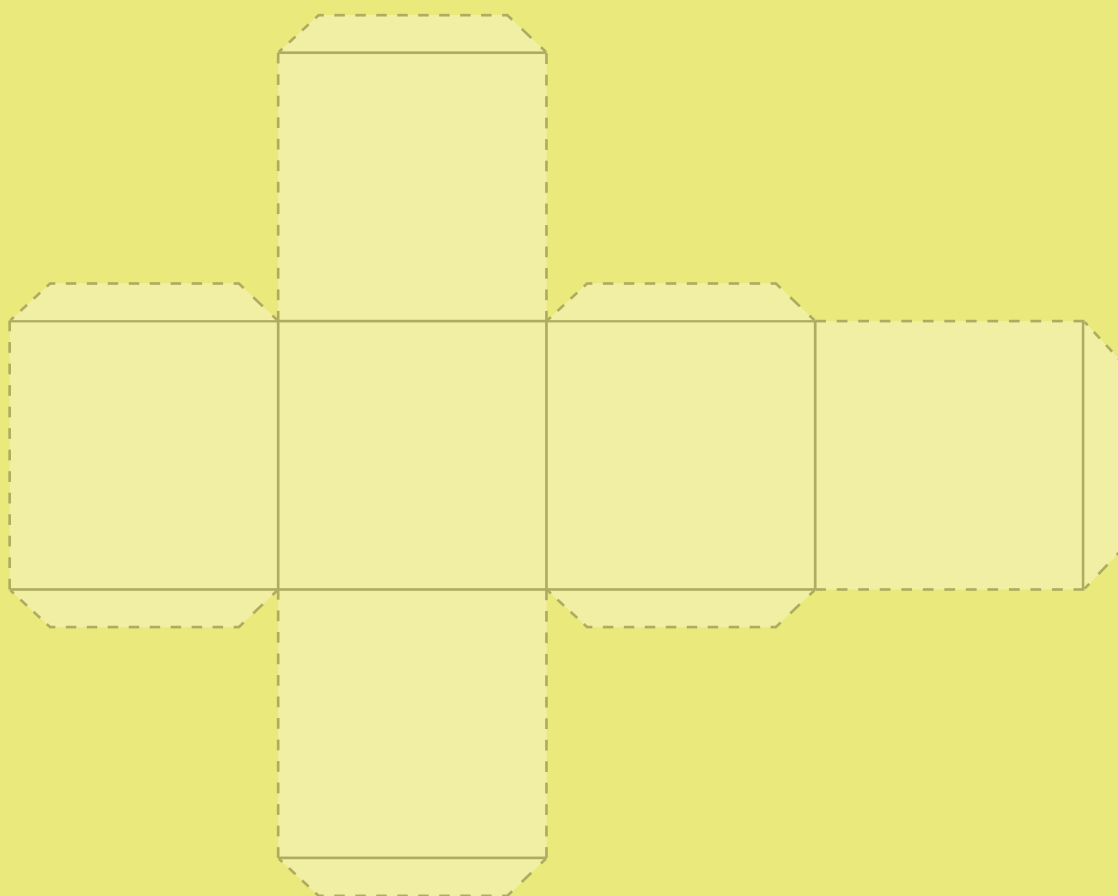
Project Manager	Assigns roles to team members; integrates their work results; and completes the game making project.
Rule Master	Sets rules which are not contradictory and which provide criteria for winning/losing.
Game Tester	Actually plays the game built so far in order to identify problems and suggest improvements.
Game Designer	Designs the physical elements of a game such as the shape of game pieces and the layout of a game board.

A Dice for My Own Board Game



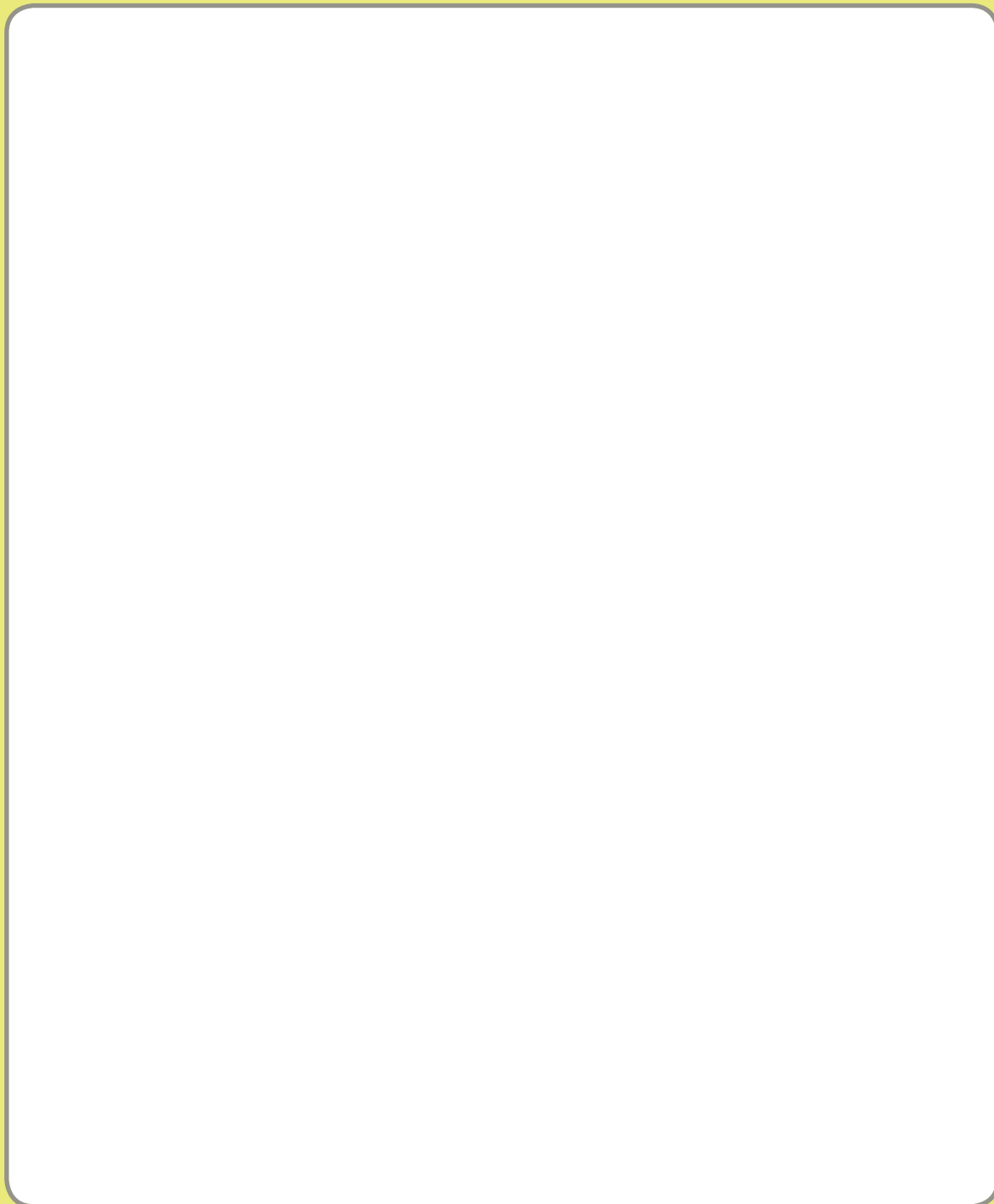
Game Pieces for My Own Board Game





A Game Board for My Own Board Game

Draw your planned game board and use it to play a game.



PART 2

Recommended target
Middle school liberal semester

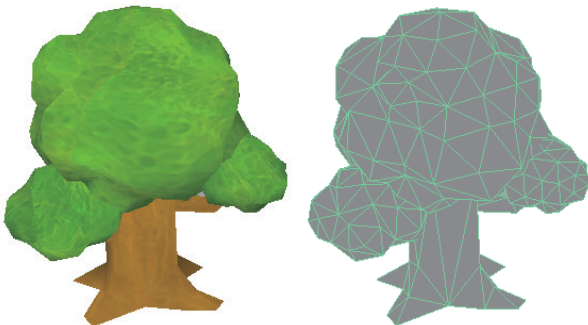
Relevant subjects
Technology & Home Life

How Game Characters Move

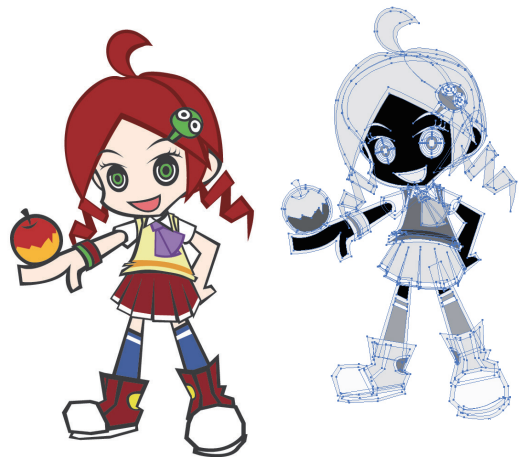
With the advancement of IT, games that are not only for computers but also for mobile devices such as smartphones and tablets are increasingly being developed. In order to make such games, you first need to know what happens in a game. A game is a form of software in which the user, devices, and game elements continuously interact interact with one another.

Game characters consisting of small geometric shapes

How are game characters expressed? Creating a game character requires modeling, rendering, and animation techniques. Modeling refers to making the basic shape of a character as a game element such as a person, tree, land, or cloud. Once modeling is done, you have to make the character look natural by brightening or darkening each part of the character and by adding a shadow, considering the direction and the intensity of the light around the character. This is called rendering. Animation refers to enabling the realistically expressed character to move.



A 3D character (polyhedral modeling)



A 2D character (polygonal modeling)

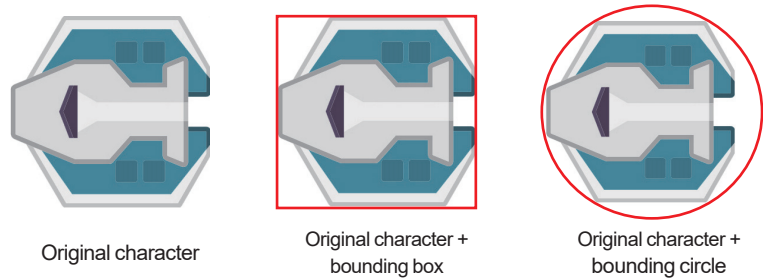
Be they people, animals, or inanimate objects, game characters are represented by geometric shapes. A 3D character is made up of polyhedrons such as spheres and hexahedrons, while a 2D character consists of dots and polygons such as triangles and quadrangles. Sometimes, curved lines and surfaces are also used to make a character look smoother. Let's assume that the item shown below appears in a game. Divide the item into triangles. How many triangles are required at a minimum?

Original item	Combination of triangles	Minimum number of triangles
		

Collision detection

Game characters and items continuously interact with one another. Your character fights against an enemy, gets an item, or detours around a rock. Because all the game objects are represented by geometric shapes as mentioned before, the interaction between two objects is in fact a computational result of two shapes colliding with each other. This kind of computational work is called “collision detection.” In order for a game to be played smoothly without stopping or slowing down, collision detection must be performed quickly and accurately.

Let's assume there are two polygons on a flat surface. The two polygons can be partly overlapping or separated from each other. Otherwise, one polygon can be inside the other one. If there are many polygons, it will take much time for even the fastest computer to calculate the number of cases perfectly. Thus, game developers have to decide between sophisticated computation and simplified/fast computation. Most developers prefer the latter.



To do this, they enclose a character or an item with a large geometric shape. For example, the planes shown above are enclosed with either a quadrangle or a circle. These shapes are called a “bounding box” and a “bounding circle,” respectively. Collision detection is performed by using bounding boxes/circles instead of complicated original characters or items because bounding boxes/circles allow for quicker and simpler computation.

Number the following situations in order from the most time-consuming to the least time-consuming.

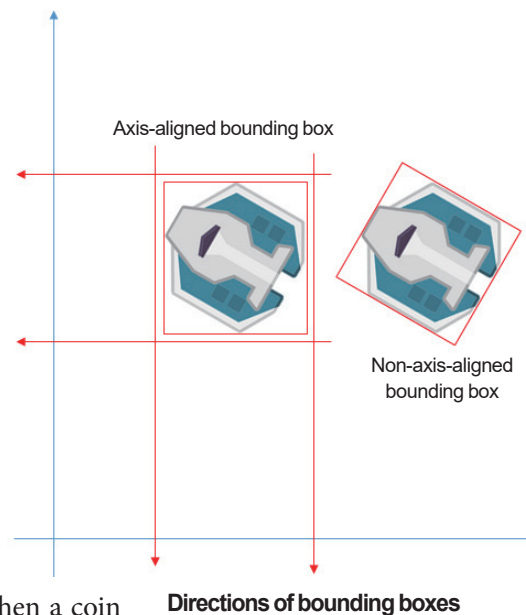
Situations			
	Two objects are obliquely placed and collision detection is based on their boundaries.	Two objects are obliquely placed and collision detection is based on their bounding boxes.	The bounding boxes of the two objects are parallel to each other.
Number			

What are the pros and cons of performing collision detection by using bounding boxes instead of the original shape of objects?

Pros	
Cons	

and third situation of the table above, it takes less time to perform collision detection when the two boxes are parallel to each other than when the boxes are placed obliquely because the former situation is much simpler. Therefore, aligning the bounding boxes of all the objects, regardless of where each object is headed, will dramatically reduce the time for collision detection.

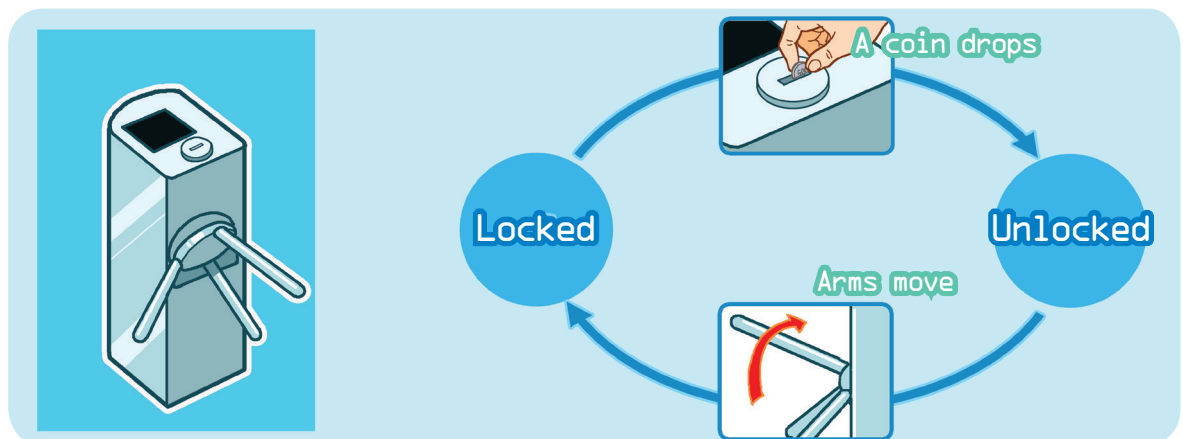
Then, how is collision detection performed for 3D games? While 2D games use bounding boxes/circles, 3D games use simple 3D shapes called a “bounding cube” or “bounding sphere.”



State diagrams

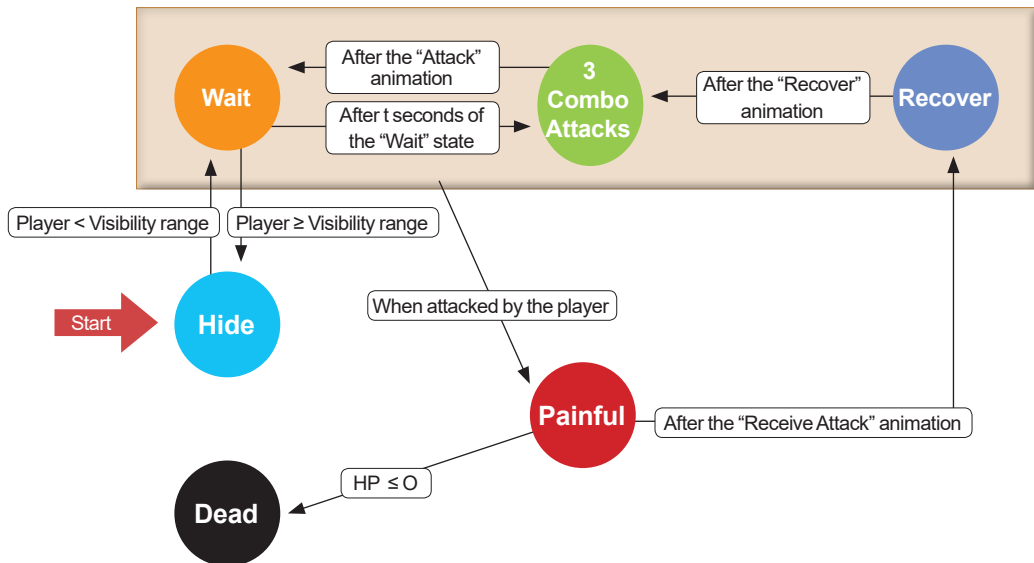
The turnstile shown in the figure below is unlocked when a coin drops into it. The turnstile is automatically locked again when its arms move. There are two states, namely, “unlocked” and “locked,” and these states are the results of the conditions “a coin drops” and “turnstile arms move,” respectively.

A diagram that shows how a game object will move under various conditions is called a “state diagram.” In order to make a program that



allows game objects to move, you have to first make a state diagram that describes conditions and motions such as “If a coin drops, the turnstile will be unlocked” and “if a character bumps into a wall, the character will take one step back.”

The figure below shows the state diagram for a monster in a game. Depending on the conditions, the monster is in the state of “Hide,” “Wait,” “Attack,” “Recover,” etc. and the corresponding motion appears in the game.



The state diagram of a monster



Initially, the monster is in the “Wait” state. When the player is within its visibility range, the monster’s state changes into the “Hide” state. After some time passes, the “Wait” state turns into the “3 Combo Attacks” state. When the monster is attacked by the player, it is in the “Painful” state. When the monster’s HP (health points) reaches zero, its state changes into the “Dead” state. In short, you can define overall behavioral patterns of the monster by defining various possible states and corresponding conditions.

Various posture and facial expressions of a monster

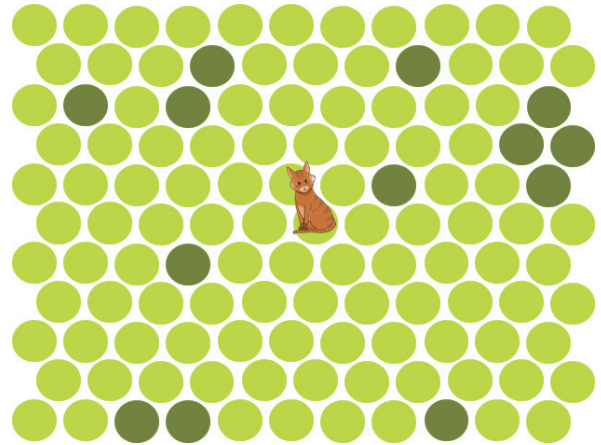
If a certain condition is met, the corresponding motion appears in the game.

Decision-making through game algorithms

Find the flash game Circle the Cat on the Internet. In this game, the cat can only move to a light green dot and you will win the game if you circle the cat with dark green dots so that the cat cannot exit the board. When the game starts, the cat sits on a light green dot. When the player clicks on one of the dots, its color turns into dark green. Then, the cat moves to another light green dot because it cannot move to a dark green dot.

In order to build this game, you have to make an algorithm to decide which dot the cat will move to. In general, the following principles are considered when making an algorithm:

1. Same results must be produced for the same situation.
2. The amount of computation shall be minimized.
3. Memory needed for computation shall be minimized.
4. The algorithm shall be efficiently maintained even if the number of dots increases.



The start screen of Circle the Cat

Deciding which dot the cat will move is different from finding solutions to the four fundamental arithmetic operations or equations. There may be more than one solution, and you have to find the most “efficient” way.

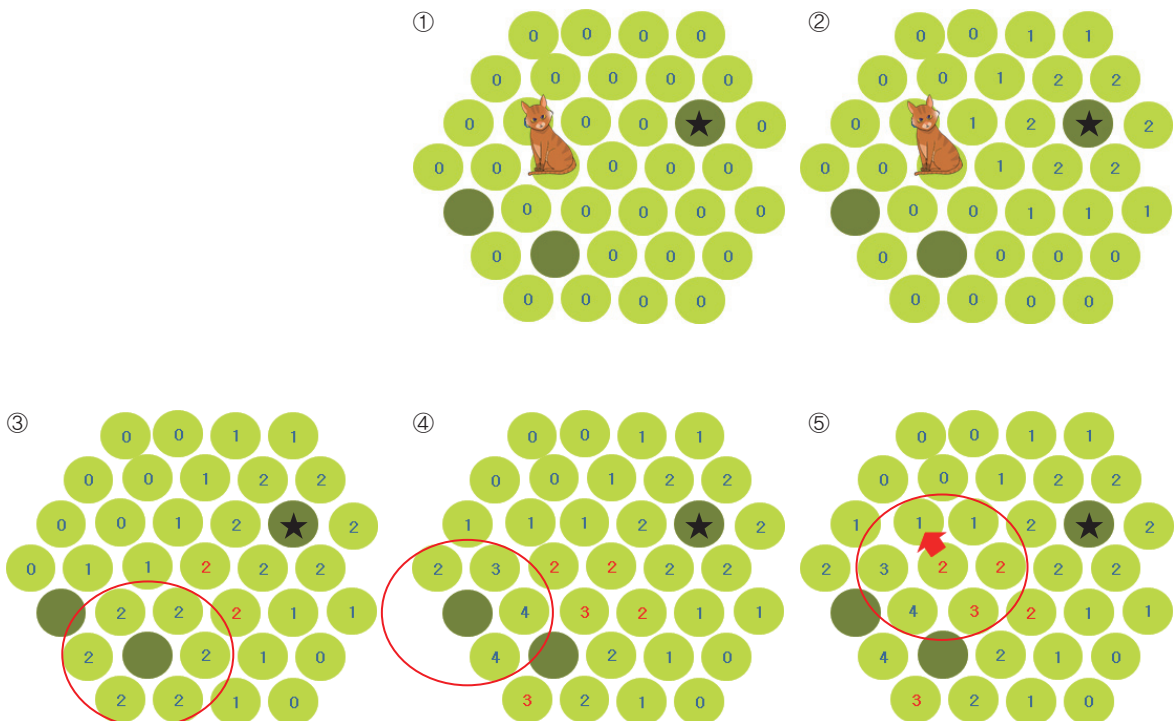
What if you make an algorithm that lets the cat randomly choose one of the possible (maximum) six dots surrounding it? Because most players have certain strategies, it is more likely that the player will win.

That being said, is it possible to decide the dot that has the lowest “risk?” In order to have the computer solve this problem, you should set criteria for assigning numerical values to all the light green dots according to the degree of risk and let the computer do the calculation. Next, you should set the program to have the cat move to the least dangerous dot.

Let's decide the risk of each dot numerically. First, assign a "0" to all the light green dots as they initially have zero risk (①). Next, assign a "2" to the six light green dots surrounding a dark green dot on the upper right corner and assign a "1" to the 12 light green dots surrounding the six dots (②).

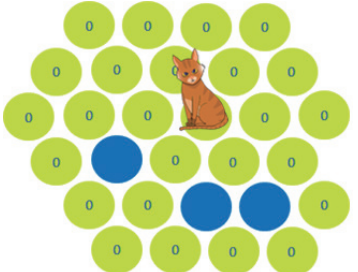
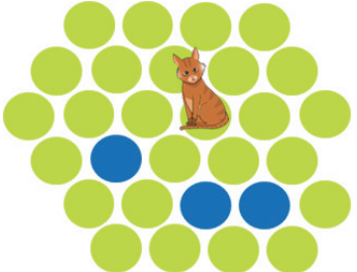
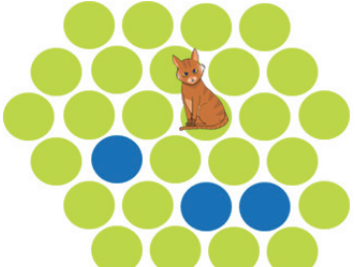
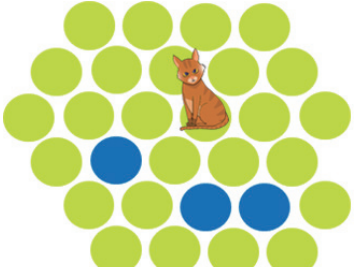
From there, assign numbers to light green dots in the same way as above, based on each of the dark green dots on the lower left corner, and calculate the total value of each light green dot (③ and ④). In this way, you can make an algorithm to let the cat move to a dot with a value of "1" among the six dots that surround it.

In this kind of algorithm, risk is calculated again whenever the player adds a dark green dot, so that the cat can move to the least dangerous dot.



Risk calculation and a decision to move based on a dark green dot on the upper right corner

Now, perform risk calculation for the situations as shown below. In the first situation, assign a “1” to the green dots surrounding each of the blue dots. In the second situation, assign a “2” to the green dots surrounding each of the blue dots and assign a “1” to the green dots surrounding each of the dots with a value of “2.”

Risk calculation (1)		
Risk calculation (2)		

Besides creating a risk calculation algorithm, you can also consider the shortest exit path for the cat. This kind of algorithm calculates all the paths to each dot on the edge and let the cat choose the shortest path. Which algorithm do you think is more like humans? In order to answer this question, you should create a program according to real algorithms and compare the results. For now, write down which algorithm you think will require more computation quantity and why.

Algorithm that needs more computation quantity	
Reason	

Activity 2

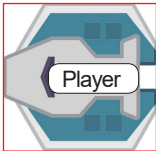
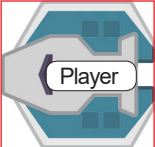
Aircraft Collision Detection

Game characters and items interact with each other by way of collision detection. The overall speed of a game depends on how fast the collision detection is performed. In order to speed up the performance, bounding boxes are used to simplify the shape of objects. This time, our activity is about the collision detection of two nearby planes.

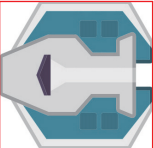
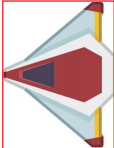
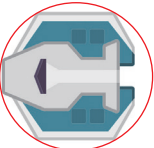
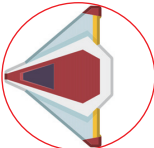
What to prepare Pencil

Activity

- In an aircraft shooting game, the player's plane loses part of its energy or the game ends when it collides with an enemy plane. If the player's plane is on the left side and an enemy plane on the right side, how will the computer calculate, by using bounding boxes, whether or not the two planes will collide?

Situations	Situations (illustration)		Mathematical explanations to avoid collision
Right (player) Left (enemy)			
Left (player) Right (enemy)			

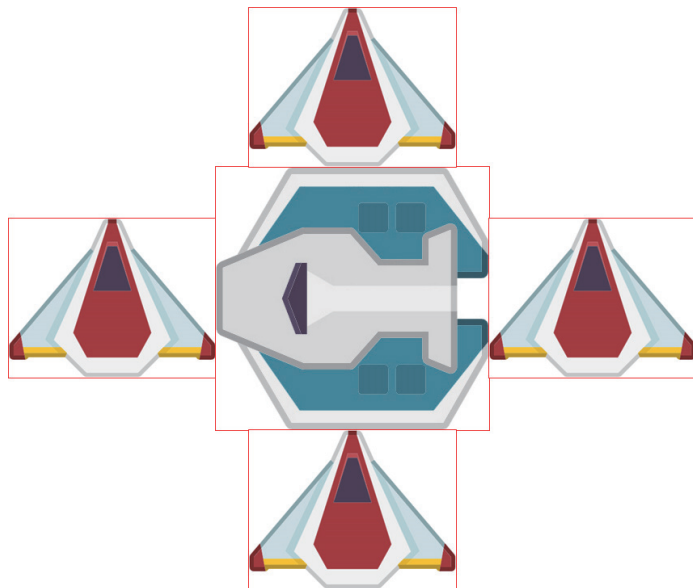
- What kind of calculations does the computer perform for the collision detection of bounding boxes and bounding circles? Tick the appropriate parentheses.

Situations	Situations (illustration)		Calculations by the computer
Collision detection of bounding boxes			Comparison of coordinate values () Distance between central points ()
Collision detection of bounding circles			Comparison of coordinate values () Distance between central points ()

3

The player's plane is about to collide with an enemy plane. Define conditions that must be met in order to avoid collision, on the basis of the positions of bounding boxes.

Plane positions



Describe a situation in which a collision would not occur, on the basis of the positions of the bounding boxes.

Translate the above description into numbers, by comparing the positions of the bounding boxes.

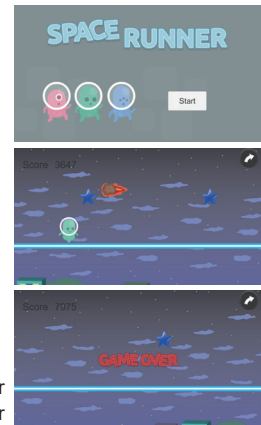
Activity 3

Making a State Diagram

A state diagram shows conditions under which game characters or items change from one state to another. This time, we will make a state diagram of a “running game.” In this kind of game, the player character has to continuously move forward while avoiding obstacles and traps. We will use a game called Space Runner.

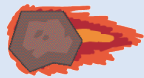
What to prepare Black/colored pencils, eraser, water-based pen

Game start, play, and game over screens of Space Runner



Activity

1 The main elements of Space Runner are as follows

Game elements	Roles	Images
Alien	layer character; jumps when the screen is touched	
Meteor	Obstacle; game ends when the character collides with a meteor	
Stars	Items for getting points	
Gem	Item for getting bonus points	

2 Summarize game rules considering the characteristics of running games and the main elements of Space Runner.

①

②

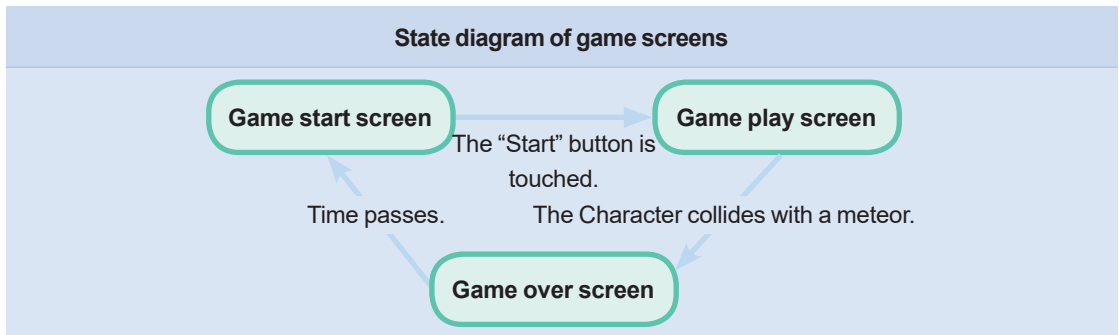
③

④

⑤

3

The following state diagram shows conditions for game start, play, and game over screens

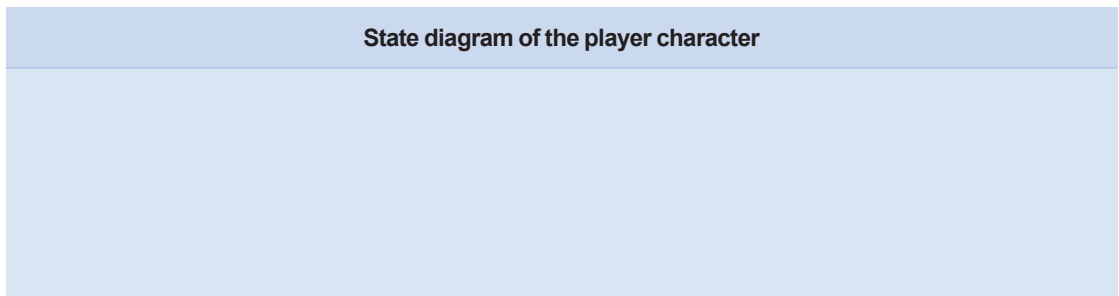


4

Considering the game rules, the player character's states are as follows.

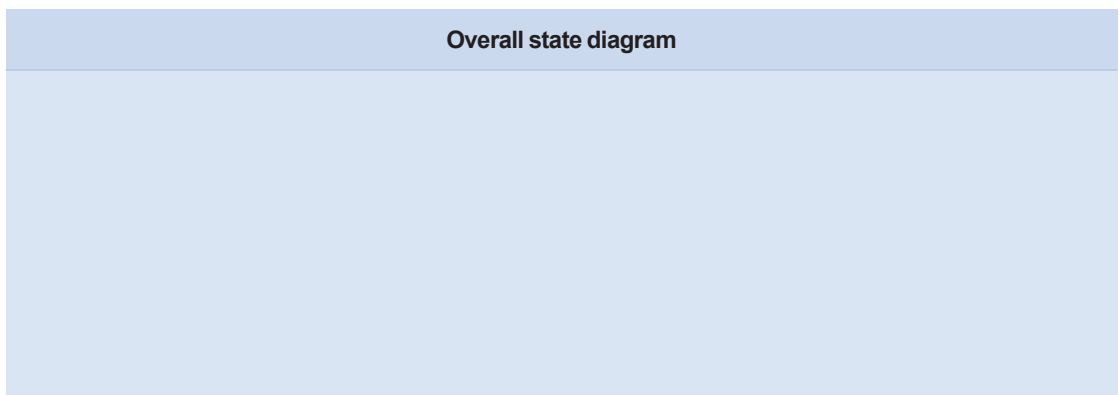
- ① The character runs. ② The character jumps. ③ The character is dead.

Of the three states, choose two. Draw a state diagram showing the conditions under which one state turns into the other. Arrows must also be shown on the diagram.



5

Merge the game screen diagram with the character diagram and complete the overall state diagram, considering other game elements such as items and obstacles as well.



PART 3

Recommended target
Middle school liberal semester

Relevant subjects
Technology & Home Life

Becoming a Game Developer

A game is a kind of a computer program, and a computer program is a package of various conditions and corresponding commands. Effective composition of commands will speed up the program execution. The framework of such game programs is called “game logic*,” while game elements including backgrounds, characters and their missions, stories etc. are called “game contents.” Game logic and game contents serve as the form and content of a game, respectively. If you change game contents while keeping the game logic, you can build a new game very quickly. It is like changing the content of your presentation with the design and the effects of the PowerPoint slides remaining intact.

* Logic

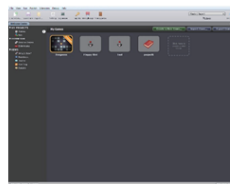
A computer performs “logical operations” which consist of certain conditions and corresponding results (e.g.: “If the character collides with an obstacle, the game will end.” “If the **Start** button is clicked on, the game will start.”). Logical operations are different from mathematical calculations such as “ $1+1=2$.” In fact, a game is played by way of numerous logical operations.

Game engine

Game logic

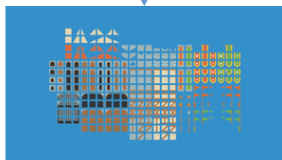


Authoring environment

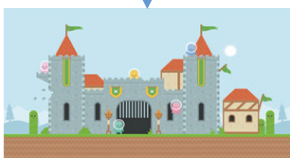


- + Support for multiple platforms
- + Support for analysis tools
- + Graphic treatment
- + Support for collaboration

Game contents A



Game A



Game contents B



Game B

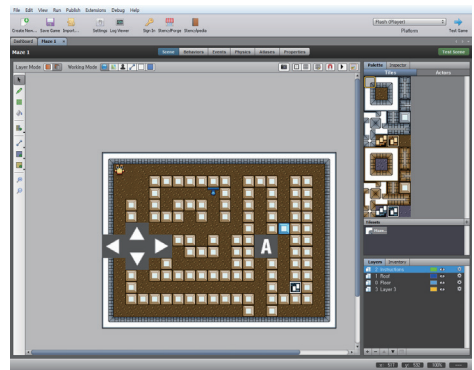


Game engines: an easier way to use game logic

A game engine is a software framework for game execution. It allows for easier use of game logic and helps developers effectively create games for multiple platforms such as iPhones and Android devices.

In order to utilize game engines effectively, you should first understand the process of making a game. You should also be familiar with the details such as collision detection and state diagrams that we have addressed before. Moreover, you should be aware of computer graphic techniques and various input methods because continuous interaction between a game and a player is shown on a screen.

Now, let's make a game by using a game engine. First, let's make a plan. Fill out each section.



Making a game by using the game engine "Stencyl"

What kind of game would you like to make by using a game engine?	
① What is the basic idea about the game?	① Flying a plane without colliding with a rock
② What game elements are necessary?	② Plane, rocks, background images, etc.
③ How do you play the game?	③ If the space key is pressed, the plane moves upward.
④ What kind of mathematical calculation will be most frequently used in the game?	④ Calculating whether or not the plane will collide with a rock
⑤ What would be the most difficult part in making the game?	⑤ It may be difficult to calculate how high the plane should move upward.

Activity 4

Building a Game by Using a Game Engine

Now, let's make a simple game by using GameSalad (www.gamesalad.com), a game engine that is widely used in the U.S. GameSalad allows people to build a game by simply configuring supported commands. Thus, even though you do not know programming languages such as C, Python, and Java, you can make a game very easily.

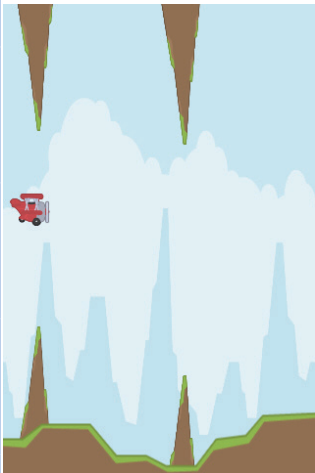
If you made a game by using GameSalad, you could actually play it at <http://arcade.gamesalad.com/>. GameSalad is for commercial games and has many features ranging from making to publishing, so at first glance making a game may look very complicated. For basic concepts, see a manual at <http://gamesalad.com/manuals>. In this activity, we will try to understand the game making process by making an obstacle avoiding game.

What to prepare A computer to install GameSalad

Activity

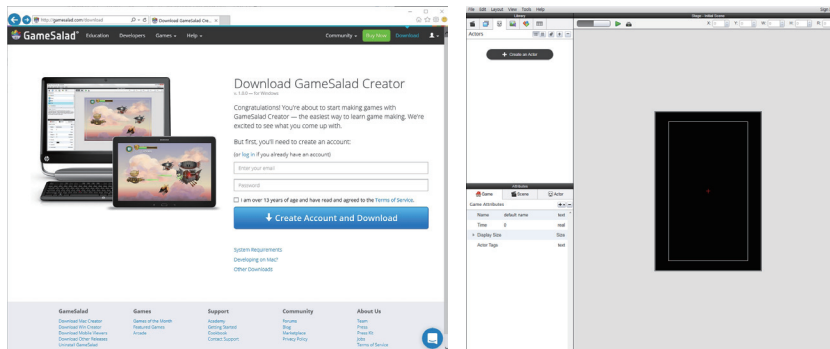
1 Think of game elements

We are trying to make a game in which a plane keeps flying without colliding with a rock. The following are necessary game elements:

Elements	Roles	Images	Game Screen
Plane	Player character; moves upward if the space key is pressed		
Top rock	A jagged rock on the top edge of the screen		
Bottom rock	A jagged rock on the bottom edge of the screen		
Ground	A screen scrolling effect makes the plane look as if it is flying.		
Background	A background image		

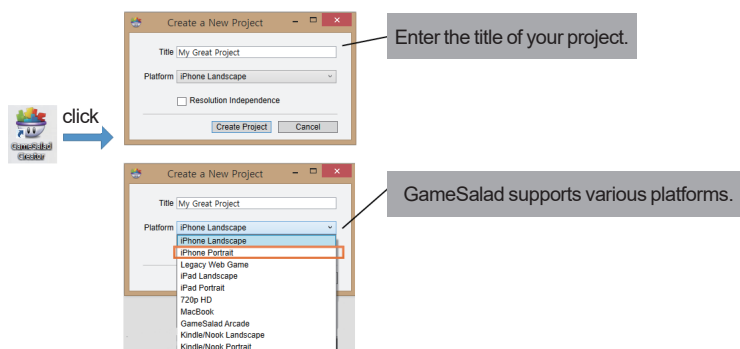
2 Install and run GameSalad

Download GameSalad at www.gamesalad.com and install it. In order to download the program, you have to subscribe first. Enter your email address and password, tick the box beside “I am over 13 years of age...,” and click on the “Download” button. If you have finished installing it, run the program.



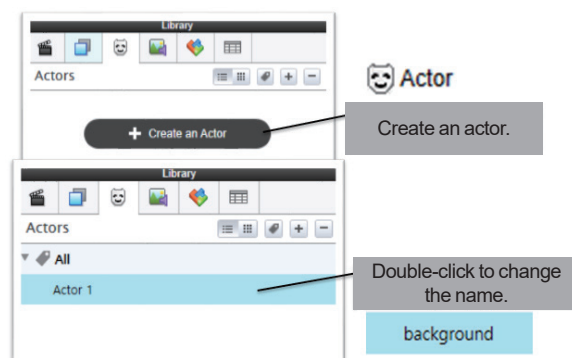
3 Create a project

Run GameSalad and create a new project. Choose the “iPhone Portrait” platform. The project screen has 320x480 pixels.



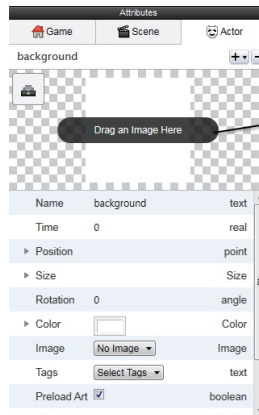
4 Create an actor

Make an actor to use it as a background. You can change the name of the actor by double-clicking on it. Change the name to “background.”

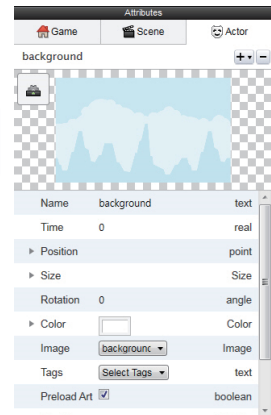


5 Insert a background image into the “background” actor

Drag a background image to the “background” actor.

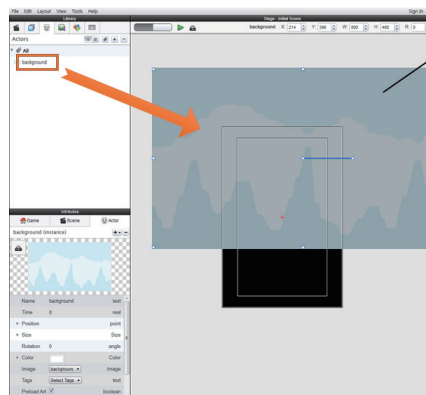


Drag a background image to the “background” actor.



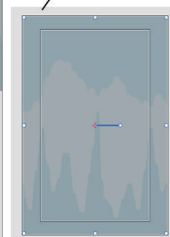
6 Adjust the background image to the screen size

Drag the “background” actor to the Scene and adjust it to the screen size.



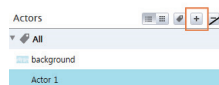
Drag the “background” actor to the Scene.

Scale up the image to fit the screen.



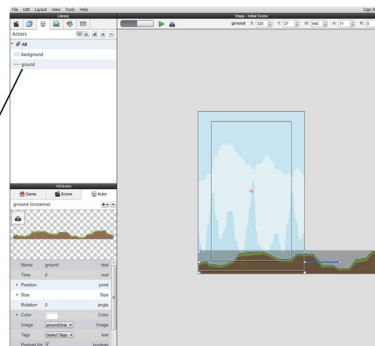
7 Insert a ground image

Create a “ground” actor and place it at the bottom of the screen.



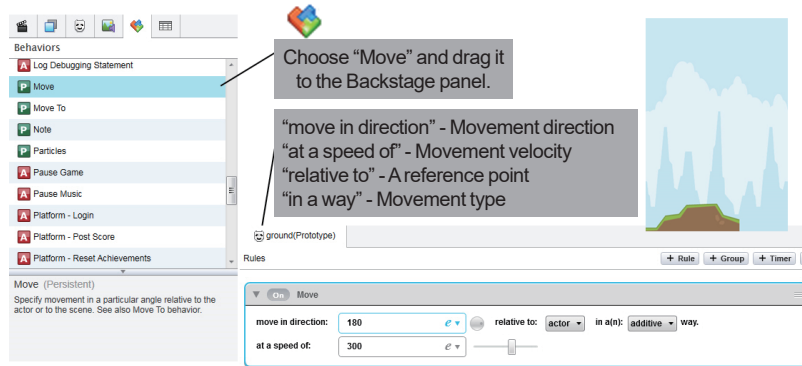
Add a new actor and name it “ground.”

Insert a image and drag the “ground” actor to the Scene.



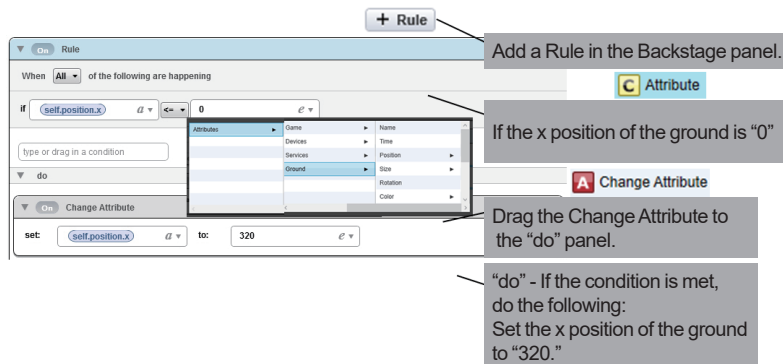
8 Move the ground image

Add a Rule to the ground image. The Move command below means the ground moves along the x axis at a speed of 300. Click on the Run button in the upper middle part of the window, and the ground will move.



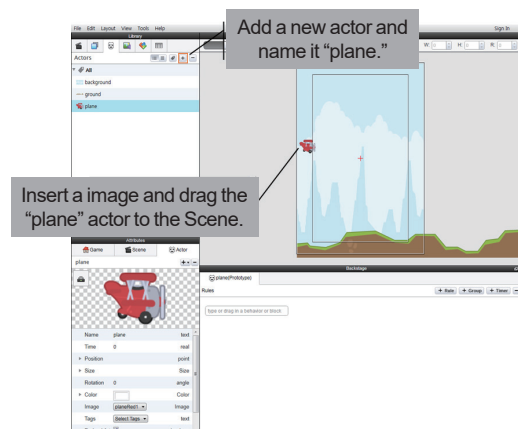
9 Make the ground move endlessly

Let the ground's position be reset to the initial x value if the x value becomes below zero. This will make the ground move endlessly.



10 Bring a plane image

Create a new actor and change the name to "plane." Insert a plane image into the actor. Drag the "plane" actor to the Scene and adjust the size of the plane.



11 Code the plane's motion

Make the plane move upward when the space key is pressed. As shown in the figure below, add a Rule and set the linear velocity. In order to make the plane drop when the space key is not pressed, add a command to apply acceleration downward.

+ Rule Add a Rule in the Backstage panel.

C Key Condition: If the space key is pressed

A Change Attribute "do" - If the condition is met, do the following:
Set the linear velocity of the plane to "150."

P Accelerate "else" - If the condition is not met, do the following:
Set the direction of acceleration.
Set the value of acceleration.

When **All** of the following are happening

receives a key press: **space** is: **down**

set: **Self.motion.linearVel** to: **150**

accelerate in direction: **270** relative to: **actor**

with acceleration: **300**

12 Create the "rock" actors

Create actors for jagged rocks ("rock_top" and "rock_bottom") and enter the Move commands so that each actor will move from right to left.

Actors

All

background

ground

plane

rock_top

rock_bottom

top

bottom

Move

move in direction: **180** relative to: **actor** in a(n): **additive** way.

at a speed of: **300**

"move in direction" - Movement direction
"at a speed of" - Movement velocity

13 Create an actor that makes rocks

Create an actor (“rock_manager”) that makes rocks at the top and bottom of the screen. Add the Spawn Actor commands so that the rocks will be created every 0.5 seconds. If you want the distance between rocks to change every time, set the y position of the rocks to “Random” within a certain range.

Create rock manager

Change Attribute

Timer to create rocks every 0.5 seconds

"in the direction" - Direction of the rocks

"relative to" - A reference point (actor)

"from position" - Position of the rocks (x axis: 320, y axis: random)

"from position" - Position of the rocks (x axis: 320, y axis: random)

14 Set conditions under which the game will end

Finally, add a Rule to the “plane” actor so that the game will end and will be reset if the actor collides with the ground, the top rock, or the bottom rock.

Rule

When Any of the following are happening

actor overlaps or collides with: actor of type ground

actor overlaps or collides with: actor of type rock_bottom

actor overlaps or collides with: actor of type rock_top

collision

do

Reset Game

reset the game

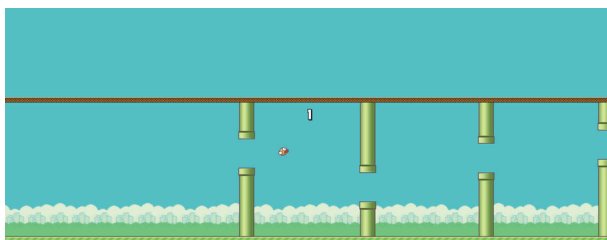
type or drag in a behavior or block

else

Activity 5

Making Flappy Bird by Using GameSalad

Flappy Bird is a mobile game developed by Vietnamese programmer Dong Nguyen. The game was released on the iOS App Store in 2013 and has gained great popularity. Flappy Bird is a simple game where the player controls a bird, attempting to fly between rows of pipes without colliding with them. Many people became too deeply absorbed into this game, and there was controversy regarding copyright issues. Consequently, the servicing of Flappy Bird stopped in less than a year after its release. Afterwards, similar games have been steadily released. In this activity, let's draw the elements of a Flappy Bird-like game and build the game by using a game engine.



Flappy Bird (source: www.amazon.com)

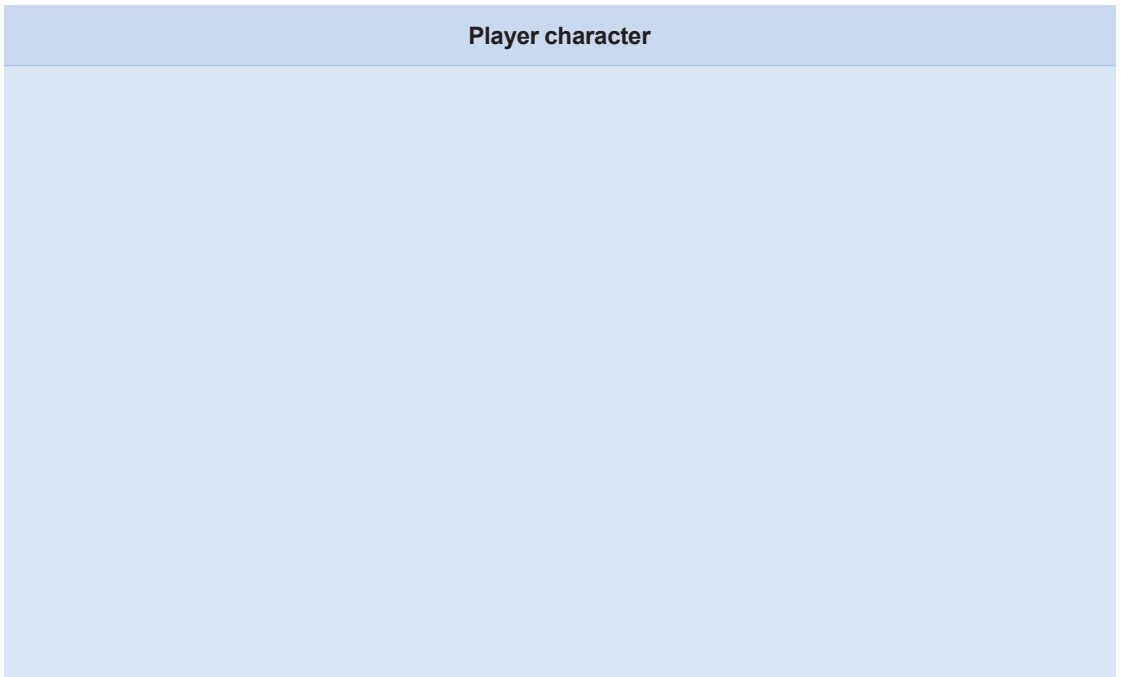
What to prepare Black/colored pencils, eraser, water-based pen, GameSalad, scanner, computer, Autodesk Pixlr (or other similar image editing software)

Activity

- 1 You are going to build a game where 'the player character moves upward/downward when the space key is pressed/released, attempting to survive as long as possible without colliding with an approaching obstacle'. Write down the details of the game.

Game title	eg) Flying Jellyfish
Background story	
How to play	

- 2** Draw the player character with colored pencils and a water-based pen. .



- 3** Draw the ground image.



4

Draw obstacles at the top and bottom of the screen.

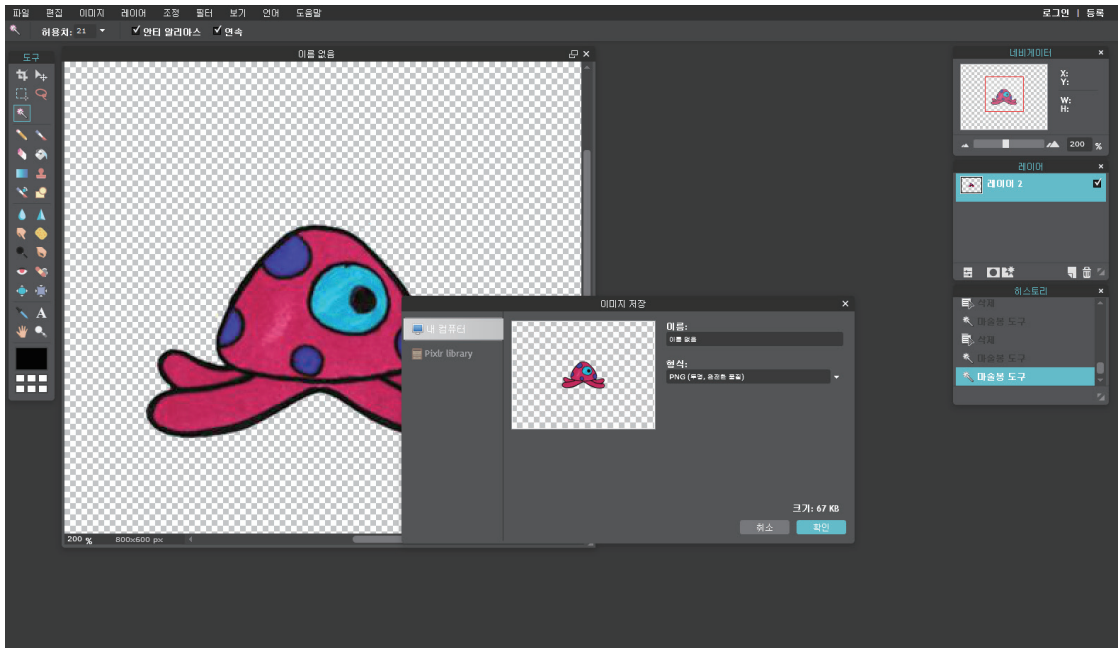
Obstacle_top	Obstacle_bottom

5

Draw the background image.

Background image

- 6** Scan your drawings and open Pixlr (<https://pixlr.com>). Make the background transparent by using the Magic Wand tool. Save the file in the “png” format.



- 7** Replace the images of the character, ground, obstacle, background, etc. of the game built in (Activity 4) with the new images, while leaving the logic part as it is. In this way, you can easily make another game of the same genre by using a game engine.

End result



Game Developer

In the contents market, games occupy the third most important position next to broadcasting and publishing. In the era of mobile technology, broadcasting and publishing are getting less popular, while the game market keeps expanding as there are various devices and ways to play games. The development of game technology necessitates many occupations such as game planning, game design, and game development. Moreover, the advancement of virtual reality (VR) technology is expected to give impetus to the game industry. For the time being, it is expected that the game industry will continue to grow in parallel with technological developments.



What do they do?

- Planning and making hardware devices for games, such as video game consoles and VR devices.
- Creating game contents such as stories, characters, and design; planning games by setting fun game rules, etc.
- Developing games by making a new game engine or using an existing one; (in the case of online games) building and maintaining a server or operating a community.

Occupations related to game development

Hardware	Game device developer (video game consoles, controllers, VR devices, etc.)
Game contents	Game scenario writer, character designer, background designer, game planner
Game development	Game developer, server manager, community operator



Related majors

Games

cartoons &
animations

computer
engineering

electric
engineering

electronic
engineering

Game planning requires more than skills for game contents: it should be backed by technological understanding in order to build games with a high standard. As games are composite art, techniques from various fields are used. Therefore, you must understand other fields as well. In order to become a game developer, you don't necessarily have to major in a related field, so long as you can acquire skills quickly.



Required aptitude

- To develop new devices, you should be able to think logically, analytically, and creatively.
- A curious and research-minded person is apt for the job. A spirit of cooperation is also needed because in most cases, you work with other people. Besides, people who are meticulous and think analytically have an advantage.
- The game development job requires an artistic sense, as well as an understanding of and interest in IT devices.



Expert interview

If you want to be a game programmer, an aptitude for mathematics is more important than anything else. Let's assume that a game character entered a cave. To make the scene look realistic, the light is essential. How does the light flow into the cave? And how is it reflected? All these questions are about mathematical calculation. Of course you also need background knowledge about science, but you have to solve equations to calculate the path of the light.

When you use a game engine such as GameSalad to build a game, you are in fact using mathematical functions made by others. If you want to be a game programmer, study mathematics hard.

Yeong-sik Park, Director of the Office of LE Development, NCSOFT



A robot catapult and a catapult game engine



Written by

Dong-man Kim (Incheon Buhyun Elementary School Teacher)



Product overview

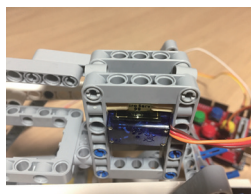
■ Product function

This device can be used to demonstrate a game engine necessary for making a catapult/cannon game. In order for a catapult to fire a payload to hit the target, you need knowledge about physics, specifically about uniform motion and falling motion. Thus, this product helps you to learn that you can make a realistic game only if the game is based on the physical properties of the objects.

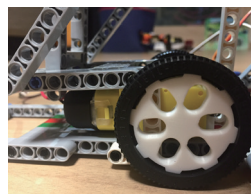
■ How it works

A servomotor enables the catapult to discharge a payload and a DC motor moves the catapult back and forth, enabling the adjustment of its distance to the target.

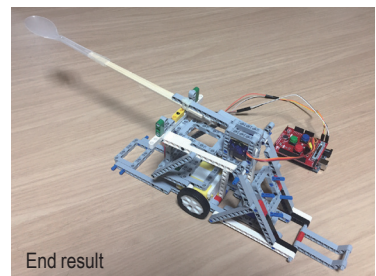
■ Product picture



Servomotor



DC motor

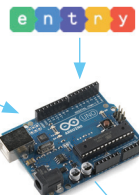


End result

■ Product structure



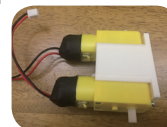
Entry sensor board
A device for digital/analog signal input by using a button and a slider. It is connected with an Arduino Uno board.



Arduino Uno board



Entry
A programming language for sending commands to the Arduino Uno board



Lego blocks
Material for making the body of the catapult

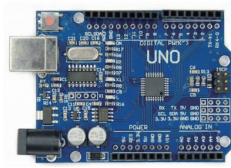


Servomotor & DC motor
Output devices that receive commands from the Arduino Uno board



Production overview

- **Production time** : 3 hours
- **Needed materials and tools**



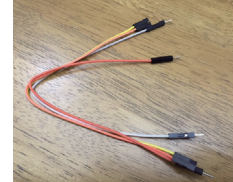
▲ Arduino Uno board



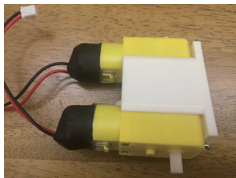
▲ Entry sensor board V2



▲ Servomotor



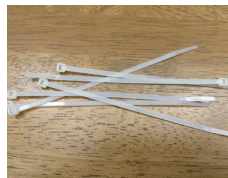
▲ Cables



▲ DC motor



▲ Lego blocks



▲ Cable ties



▲ Disposable chopsticks/spoon

■ Key production principles

Connecting an Arduino Uno board to operate a robot catapult

Load the catapult and press the button to operate the servomotor. The catapult's arm swings 90 degrees clockwise and a payload is fired. Measure the horizontal distance of the payload fired. Connect a DC motor to let the catapult move back and forth. The firing distance depends on the operational angle of the servomotor. Adjust the catapult's position and the servomotor angle for precise targeting. Make a game engine by coding the catapult's movement and firing scenes using Entry. Run the game engine and compare the firing scenes on the screen with the actual movement of the catapult and the trajectory of the payload.

■ What requires attention in production

- ① Make the catapult's arm long enough and prevent recoil when firing.
- ② Install the servomotor after adjusting its operational angle to zero degrees.
- ③ When using Entry, the Arduino Uno board must be connected to the computer via a USB cable, and power (from a battery and/or a USB port) must be constantly supplied to the catapult system.

SW Education Module Textbook

- ❶ Artificial Intelligence
- ❷ Driverless Vehicles
- ❸ IoT(Internet of Things)
- ❹ Virtual Reality
- ❺ CRISPR
- ❻ Space Launch Vehicles
- ❼ Natural Disasters
- ❽ Smart Medicine
- ❹ Game Engines**
- ❿ Sports Statistics

SW Education Module Textbook ❹ Game Engines

Game Engines from A to Z

Issued on February 23, 2017
Issued by Ministry of Science and ICT, Korea Foundation for the Advancement of Science and Creativity
Written by Gu-hwan Oh
Planned by Ji-young Park, Jong-geol Son, Hye-ja Kim (MSIP)
Yun-jeong Kim, Se-man Park, Ok-su Hong, Jae-min Han, Hye-ran Go (KOFAC)
Edited by Min-seok Lee, U-cheon Jeon
Reviewed by Jun-byeol An

Published by Doo-hui Kim
Publisher Dongascience Inc.
Responsible editor Kyeong-ae Chang

Inquiries Software Education Center, Korea Foundation for the Advancement of Science and Creativity
Email swedu@kofac.re.kr
Phone (02)559-3841~9
Address (135-867) 602 Seolleung-ro, Gangnam-gu, Seoul (Samreung Building, Samseong-dong)
Homepage www.kofac.re.kr

Computational Thinking Map

1. Traditional Games Revisited

Activity 1



Problem Decomposition

Collect information about games similar to Yut Nori



Recognizing Patterns

Compare the games similar to Yut Nori



Abstraction

Make a game board and define the composition of the board



Automation

Assign symbolic meanings to game pieces and express them as a drawing



Making an Algorithm

Define stations to be included in the game board and characterize each station

3. Becoming a Game Developer

Activity 4



Problem Decomposition

Look into the games built by using a game engine



Recognizing Patterns

Come up with a game idea and compare it with similar games



Abstraction

Draw game characters, backgrounds, and obstacles



Automation

Program a game that uses a game engine, by combining images and game logic



Making an Algorithm

Check whether the game elements were designed according to the background story of the game

Activity 2



Problem Decomposition

Look into the types of geometric shapes



Making an Algorithm

Define collision conditions in each of the situations



Recognizing Patterns

Find out how to mathematically calculate the collision between two dots and between two lines



Abstraction

Understand the 2D coordinate system and mark relative positions by using the variables x and y

2. How Game Characters Move

Activity 3



Recognizing Patterns

Choose an ordinary object and draw its state diagram



Abstraction

Learn how to express the transition of states



Problem Decomposition

Find an ordinary object (such as a vending machine) that can be expressed as a state diagram



Making an Algorithm

Make state diagrams for a game character, a game screen, and a game play



Automation

Conceptually explain whether or not an object's characteristics are easily expressed as a state diagram

software

education module



Ministry of Science and ICT



KOFAC
Korea Foundation for
the Advancement of Science & Creativity
한국과학창의재단