qiita.com

HyperCard(HyperTalk)でFlappy Birdを作る

oinariman

11–13 minutes

This is an article in the minor language Advent Calendar 2014, and it is also the third in the series "Make Flappy Bird" series. We will introduce the scripting language HyperTalk while making Flappy Bird-style games using HyperCard, which was included free of charge on the former Macintosh.

- Make a flappy bird with Sprite Kit
- Making Flappy Bird with Sprite Kit (Swift version)

What is HyperCard (HyperTalk)?

HyperCard is "the first commercial software to achieve hypertext" (HyperCard (Wikipedia). HyperTalk is a scripting language that worked on it.

Read this article about the significance and location of HyperCard in the history of computers. In terms of actual use, in a sloppy parable,

PowerPoint + Paint Tools + Scripting Language (HyperTalk)

It's like that. Like the slides in PowerPoint, each screen consists of a series of "cards" (called "stacks"). You can draw a picture with the paint tool on the card, or you can place a "field" that displays the text or a "button" that can be skipped to another card when you click. You can also use HyperTalk to give you a behavior other than transitioning to another card to a button.

HyperCard as a game development environment

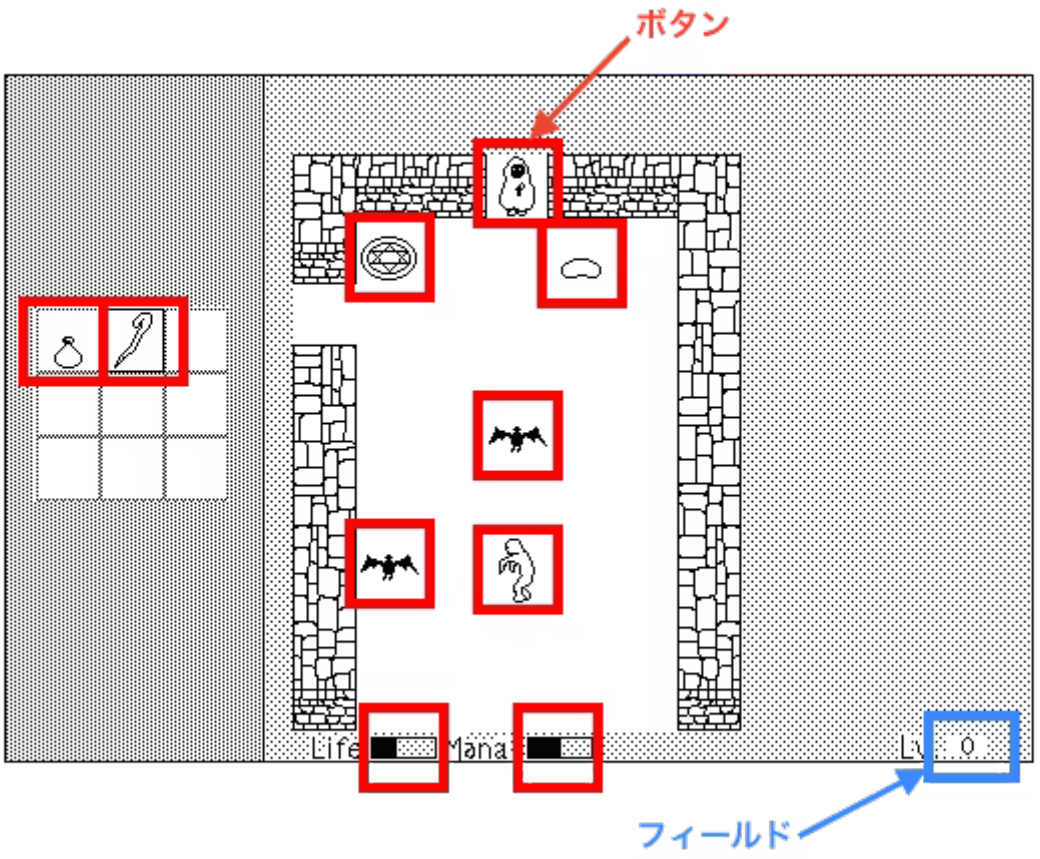
It is also introduced on Wikipedia, but Cyan's game "MYST" was made with HyperCard. MYST is an adventure game that explores an uninhabited island made of 3D graphics. At the time of the announcement, I expressed a high-definition 3D world that could not be rendered in real time with a general personal computer by a method that transitions the stop-painting of scene scenes. Since the 1980s, Cyan has been presenting children's adventure games such as The Manhole, Cosmic

Osmo and the Worlds Beyond the Mackerel, and Spellunx. Some of the graphic expressions that make full use of the black and white dot pattern of these works are eye-catching. (In the following video, I think that "Spelunx" is clear and easy to understand.)

- The Manhole (1988)
- Cosmic Osmo - Gameplay
- Spellunx and the Caves of Mr. Seudo

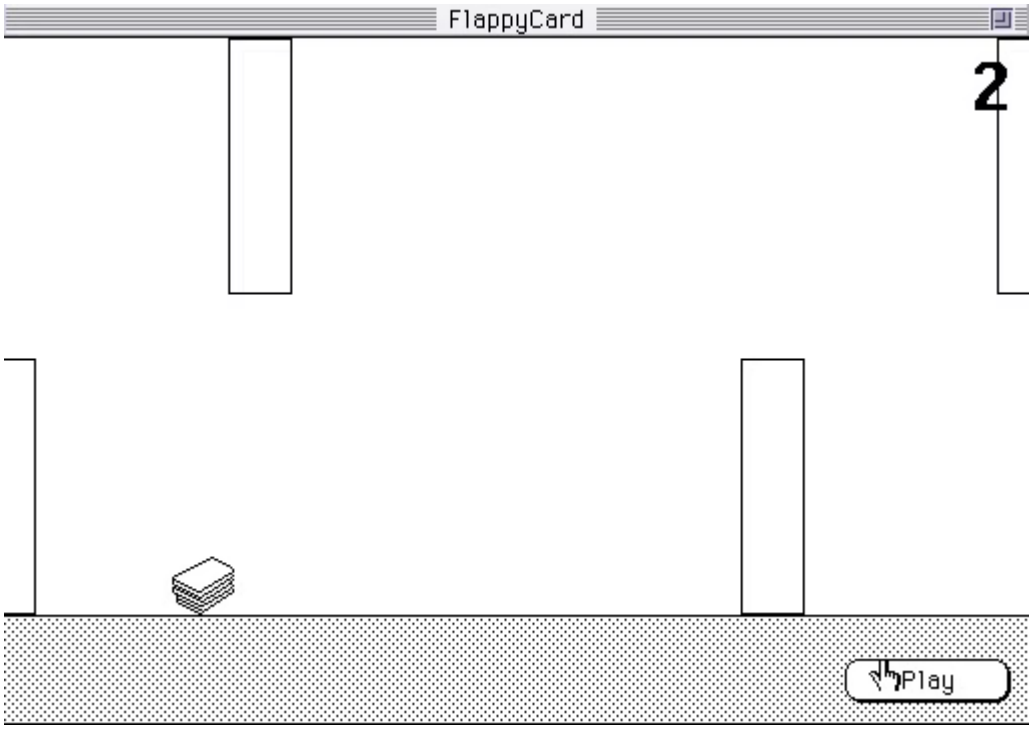
In HyperCard, you can create a gamebook-like adventure game without a script. Draw a picture of the scene on the card, and place a transparent button in the clickable part. The button can be set to the transition to another card only with the mouse operation.

Next, try to make a 2D game like Flappy Bird. Unlike an adventure game, you need a character to fly and bounce. Buttons can be used as a GameObject to replace Sprites. You can set the 32x32 pixel icon on the button. Icons can be changed in scripts and animated. Fields can be used to display scores, etc. You can use the picture drawn on the card as the background of the game. In addition, there is the concept of "background" where you can share button field backgrounds between multiple cards.



Make a Flappy Bird

Let's get into the subject. For more information on how to configure the Flappy Bird game, please refer to the article (making Flappy Bird with Sprite Kit). This time it is HyperCard, so I will try to skip the icon of the stack instead of the bird. The following results are completed:



SOURCE

It's a script for the whole game. Insert it into the "Play" button at the bottom right. Start the game when you click on it, and stop when the game is over.

```
on mouseUp
  -- initialization
  put 0 into vy
  put 1.5 into ay
  put -10 into impact
  put -8 into pillar_vx
  put 0 into count
  put false into mouse_down
  put 16 into interval
  put 5 into pmax
  put 2 into w
  put false into game_over
  repeat with i = 2 to pmax
    hide btn i
    set rect of btn i to 0,0,0,0
  end repeat
  put 0 into card field 1
```

```
set loc of btn "stack" to 100,10

repeat
  -- pillar
  if game_over is not true then
    if count mod interval is 0 then
      repeat with i = 2 to pmax
        if visible of btn i is false then
          show btn i
          if random(10) > 5 then
            set rect of btn i to 512,160,512+32,288
          else
            set rect of btn i to 512,0,512+32,128
          end if
          exit repeat
        end if
      end repeat
    end if
    repeat with i = 2 to pmax
      if visible of btn i is true then
        if item 1 of loc of btn i < -32 then
          hide btn i
          put card field 1 + 1 into card field 1
          play "boing" "a"
        else
          get loc of btn i
          put item 1 of it + pillar_vx into item 1 of it
          set loc of btn i to it
        end if
      end if
    end repeat
  end if

  -- user input
  if game_over is not true then
    if the mouse is down then
      if mouse_down is not true then
        put true into mouse_down
```

```
      put impact into vy
    end if
  else
    put false into mouse_down
  end if
end if

-- gravity
put vy + ay into vy
get loc of btn "stack"
put item 2 of it + trunc(vy) into item 2 of it
set loc of btn "stack" to it

-- collision
repeat with i = 1 to pmax
  put left of btn "stack" into l
  put top of btn "stack" into t
  put right of btn "stack" into r
  put bottom of btn "stack" into b
  repeat with j = 1 to 4
    if j is 1 then
      get l&","&t
    else if j is 2 then
      get r&","&t
    else if j is 3 then
      get r&","&b
    else if j is 4 then
      get l&","&b
    end if
    if it is within rect of btn i then
      if game_over then
        if i is 1 then
          exit mouseUp
        end if
      else
        play "boing"
        put true into game_over
        put 0 into vy
```

```
          end if
        end if
      end repeat
    end repeat

    wait w
    put (count + 1) mod 360 into count
  end repeat
end mouseUp
```

Scenes and Characters

This time I only used one card. The ground is the ground where it is painted with the pattern at the bottom of the screen. This time, we omitted the ground scrolling to represent the advance. In addition, the following parts were prepared.

Name	Type	Number	Role
-	Button	1	For ground collision detection
-	Button	2	Pillar
-	Button	3	Pillar
-	Button	4	Pillar
-	Button	5	Pillar
"The Play"	Button	6	Start game button
"The Stack"	Button	7	Bird
-	Field	1	Score display

Buttons and fields are numbered the more newly created. The larger the number, the more it will be displayed on the front, so the bird made the maximum number. The first button for determining a ground collision is transparent. I would like to assign a picture of a earthen tube to the buttons of 2 to 5 pillars, but since images of 32x32 pixels or more cannot be displayed (*), I made it just a square button as a compromise.

* Using a utility called ResEdit, embedding a PICT resource in the stack, and attaching it to a button to display it, there is also a way to display it. (See also: Speed up HyperCard)

Scenario

If you write an overview of the game loop in the BASIC style, it will look like this:

```
10 Initialization
20 Start loop
30 Create pillar
40 Move pillar
50 +1 point if pillar goes off screen
60 Jump bird when mouse is clicked
70 Apply gravity to bird
80 Collision check. Exit loop if collision occurs
90 Return to 20
```

I will explain part of the process with a pinch.

Gravity on birds.

Here is the place that gives gravity to birds: The break of the sentence of HyperTalk is a line break.

```
   -- gravity
   put vy + ay into vy
   get loc of btn "stack"
   put item 2 of it + trunc(vy) into item 2 of it
   set loc of btn "stack" to it
```

1st line

"--" means a comment line.

2nd line

Hypertalk is a grammar that is close to English. To assign a variable, use put-into, not the = operator. vy is the Y-direction speed of the bird, and ay is the Y-direction gravitational acceleration.`put vy + ay into vy`Gravitational acceleration is added for each frame. In writing in Japanese,vy+ayをvyに入れろIs it like that?

Third line.

In addition, there is "it" as an element that seems to be English.`get loc of btn "stack"`"What are you doing?" loc" is an abbreviation of the button's property

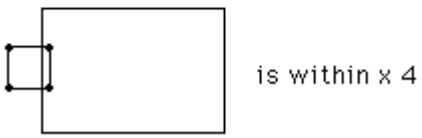
"location", "btn" is an abbreviation of "button", if written without abbreviated`get the location of button "stack"`It will be.`"stack"`という名前のボタンの位置を得よLike I said, “Get it.” This statement implicitly assigns the obtained value to the variable "it". In the following two lines, this it is used to cause the displacement of the button.

Lines 4 and 5

The location of the button is represented by the comma-separated data of the X and Y coordinates. The comma-separated data is called an "item list", and each element can be accessed with the "item" keyword. The position information of the "came" button in the third lineX座標,Y座標Because it is in form,`item 2 of it`You can get the Y coordinates. Add the Y-direction speed to it and re-introduce it to it. The coordinates must be integers.`trunc ( )`So I'm casting up. The "set" keyword on the fifth line changes the position of the button.

Collision judgment

HyperCard has a "is within" keyword that can be used to determine collisions. This determines whether a two-dimensional point is in the rectangular area. The button property "location" is the central coordinate of the button rectangle area. If you judge by the point in the center of the bird, the bird will be tucked into the wall and it will not be good, so the judgment process is performed on the four corners of the button.



The part of the collision detection with obstacles (ground, pillar) is as follows.

```
  repeat with i = 1 to pmax
    put left of btn "stack" into l
    put top of btn "stack" into t
    put right of btn "stack" into r
    put bottom of btn "stack" into b
    repeat with j = 1 to 4
```

```
      if j is 1 then
        get l&","&t
      else if j is 2 then
        get r&","&t
      else if j is 3 then
        get r&","&b
      else if j is 4 then
        get l&","&b
      end if
      if it is within rect of btn i then
        if game_over then
          if i is 1 then
            exit mouseUp
          end if
        else
          play "boing"
          put true into game_over
          put 0 into vy
        end if
      end if
    end repeat
  end repeat
```

Put the coordinates of the four corners of the bird button in the variable in advance,`repeat`Make a judgment with each obstacle in the loop.

```
    put left of btn "stack" into l
    put top of btn "stack" into t
    put right of btn "stack" into r
    put bottom of btn "stack" into b
```

`get l&","&t`The comma-separated data of the coordinates is created and assigned to the "it" variable. When a collision is detected, "play" sounds and put it in the game over state." "boing" is a default sound resource that can be used. It is also possible to play a melody by giving a scale to the second argument. This game is also used to get points:

This sounds the "A" sound, "La".

In conclusion

When I tried it, I was able to make a flappy bird. Let's make a game with HyperCard

that allows you to create intuitively interactive content! In fact, you can still get HyperCard on Apple's site, so you can run it with an old real machine or emulator!

The HyperCard 2.2 Lite-J link has disappeared from the Apple site.

> I want to download HyperCard 2.2 Lite-J.
> Can I download it now?

> It seems that it was available to download until about 7 years ago, but it is no longer available for download.