

Flying Hamster

I C T 융 합 학 부 이 수 진



Contents

01 동기 및 소개

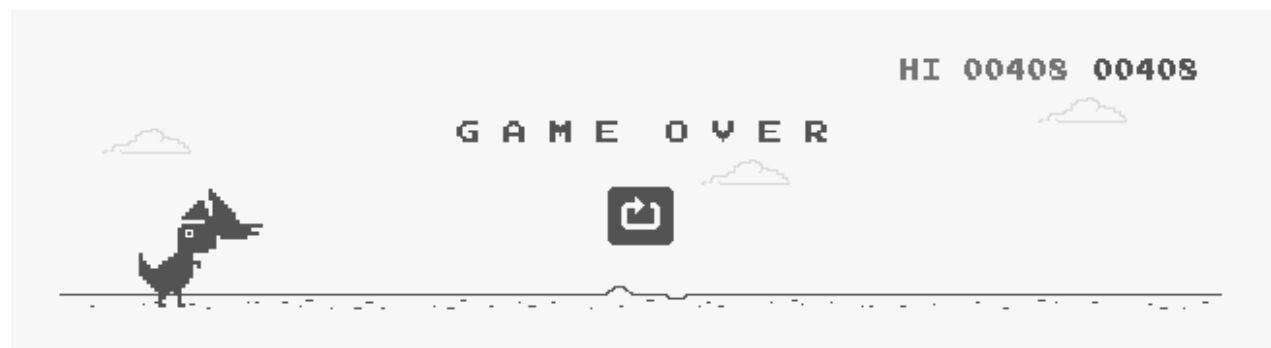
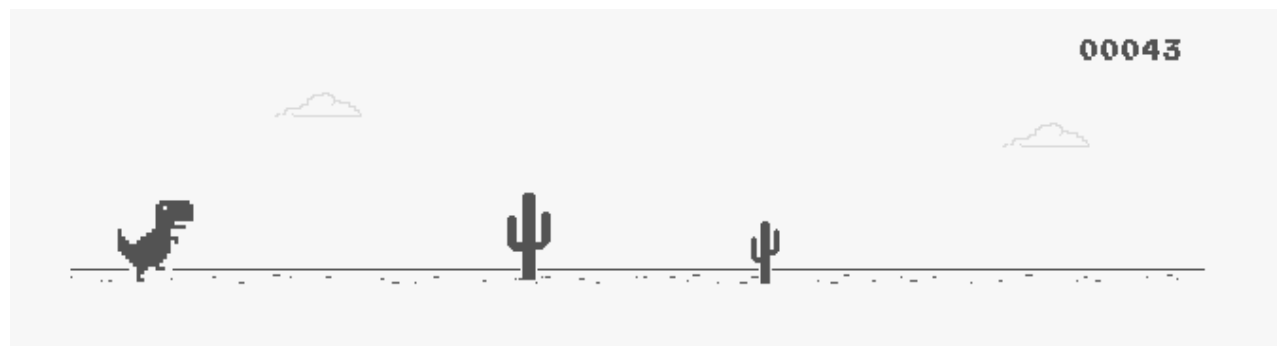
02 코드 설명

03 시연

04 보완할 점

1

동기





햄스터

hamster



바위

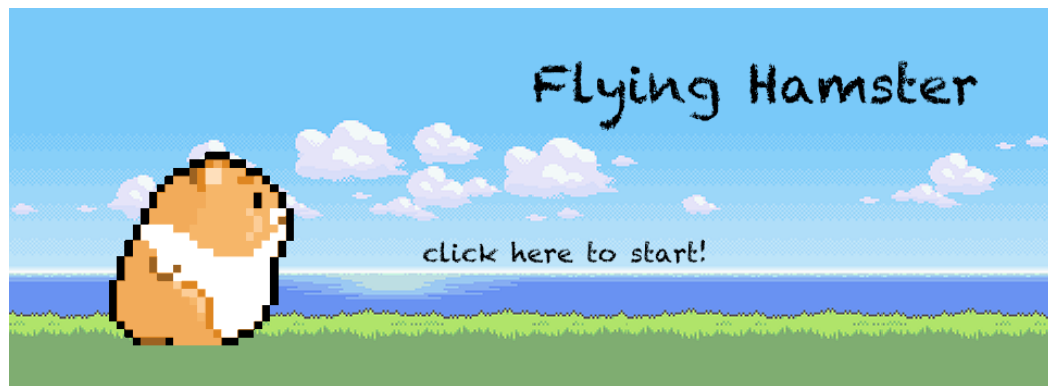
rock



귀신

ghost

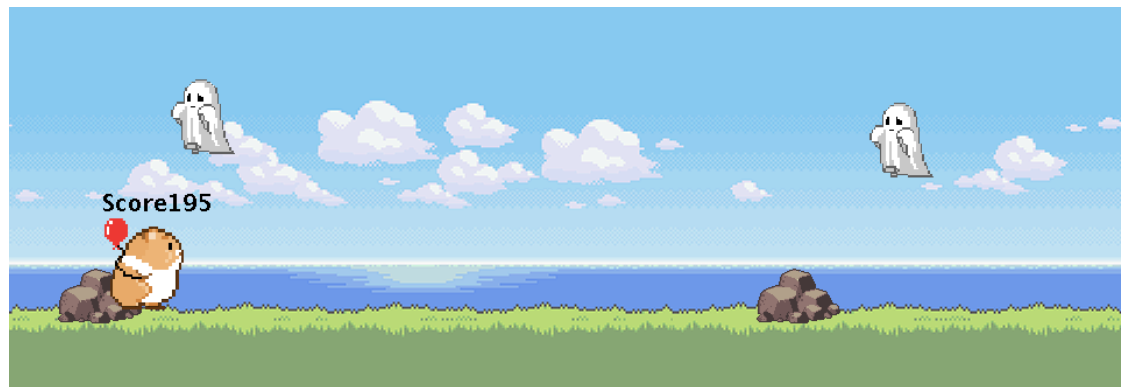
시작 화면



종료 화면

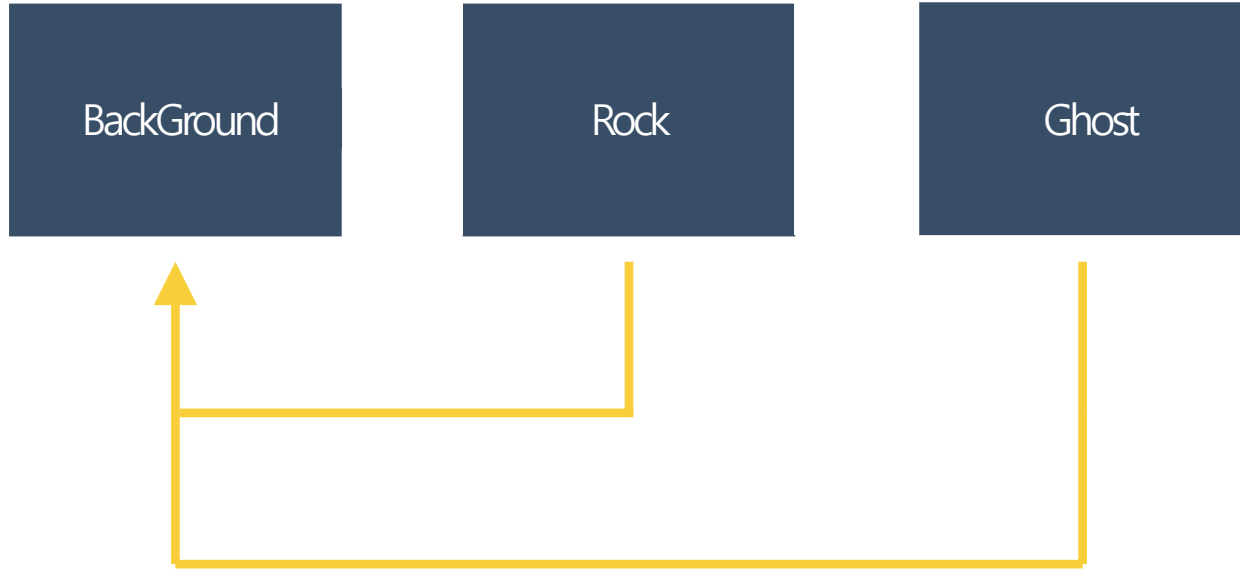


실행 화면



2

코드 설명



```

55 public void init() {
56     //image 생성
57     background_img = new ImageIcon("BackGround.png").getImage();
58     rock_img = new ImageIcon("rock.png").getImage();
59     gh = Toolkit.getDefaultToolkit().createImage("Ghost.gif");
60     image = Toolkit.getDefaultToolkit().createImage("hamm.gif");
61     jh = Toolkit.getDefaultToolkit().createImage("jumpham.gif");
62     x=80;
63     y=220;
64
65     h_w = ImageWidthValue("hamm.gif")-15;
66     h_h = ImageHeightValue("hamm.gif")-15;
67
68     g_w = ImageWidthValue("ghost.gif")-15;
69     g_h = ImageHeightValue("ghost.gif")-15;
70
71     r_w = ImageWidthValue("rock.png")-15;
72     r_h = ImageHeightValue("rock.png")-15;
73 }
74
75 public void start() {
76     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
77     addKeyListener(this);
78     cnt = System.currentTimeMillis();
79     t1.start();
80 }
81
82 public void run() {
83     try {
84         while(re==0) {
85             score=(int) ((int) (sc-cnt)/100.0);
86             KeyProcess();
87             GhostProcess();
88             RockProcess();
89             repaint();
90             t1.sleep(10);
91             count ++;
92         }
93         System.out.println("finish");
94         //buffg.drawImage(background_img, 0, 0, null);
95         Draw_End();
96         t1.sleep(100000);
97         t2.sleep(100000);
98     }catch(Exception e) {};
99 }
100

```

```

315 public Background() {
316     ImageIcon intro_img = new ImageIcon("Intro.png"); //start display
317     f_width = 960;
318     f_height = 351;
319     setSize(f_width, f_height);
320     setFocusable(true);
321
322     JButton button = new JButton(intro_img);
323     button.setText("Start");
324     button.setBorderPainted(false);
325     button.setContentAreaFilled(false);
326     button.setOpaque(false);
327     add(button);
328
329 ActionListener listener = new ActionListener() {
330     button.addActionListener(listener);
331
332     //pack();
333     setResizable(false);
334     setVisible(true);
335
336     System.out.println("?");
337
338 }
339
340 ---

```

```
101 public void keyPressed(KeyEvent e) {  
102     if (e.getKeyCode() == KeyEvent.VK_UP)  
103         //System.out.println("UP");  
104         KeyUP=true;  
105 }  
106  
107 public void keyReleased(KeyEvent e)  
108 {  
109     if(e.getKeyCode() == KeyEvent.VK_UP)  
110         KeyUP= false;  
111 }  
112  
113 public void keyTyped(KeyEvent e) {  
114 }  
115  
116 public void KeyProcess() throws InterruptedException{  
117     //System.out.println("KeyProcess");  
118     int up=0;  
119  
120     if (KeyUP==true) {  
121         //System.out.println("UP true");  
122         y-=5;  
123     }  
124 }  
125
```

```

154 public void GhostProcess() throws InterruptedException {
155     for(int i = 0; i<Ghost_list.size(); ++i) {
156         ghost = (Ghost)(Ghost_list.get(i));
157         ghost.move(score);
158         if(ghost.x<-960) {
159             Ghost_list.remove(i);
160         }
161     }
162     if(score>700) {
163         if(count%80==0) {
164             ghost = new Ghost(f_width+100,y);
165             Ghost_list.add(ghost);
166         }
167     }
168     else if(score>300) {
169         if(count%100==0) {
170             ghost = new Ghost(f_width+100,y);
171             Ghost_list.add(ghost);
172         }
173     }
174     else if(score>150) {
175         if(count%200==0) {
176             ghost = new Ghost(f_width+100,y);
177             Ghost_list.add(ghost);
178         }
179     }
180     else {
181         if(count%300==0) {
182             ghost = new Ghost(f_width+100,y);
183             Ghost_list.add(ghost);
184         }
185     }
186     for(int j = 0 ; j< Ghost_list.size(); ++j) {
187         System.out.println(Ghost_list.size());
188         ghost = (Ghost)Ghost_list.get(j);
189         if(Crash(x,y,ghost.x,ghost.y,h_w,h_h,g_w,g_h)) {
190             re++;
191         }
192     }
193 }
194 }
195 }
196 }
197 }

```

```

234 public void RockProcess() {
235     for(int i =0; i<Rock_list.size(); ++i) {
236         rock = (Rock)(Rock_list.get(i));
237         rock.move();
238         if(rock.x<-960) {
239             Rock_list.remove(i);
240         }
241     }
242     if(count%300==0) {
243         rock = new Rock(f_width+100, 245);
244         Rock_list.add(rock);
245     }
246     for(int j = 0 ; j< Rock_list.size(); ++j) {
247         System.out.println(Rock_list.size());
248         rock = (Rock)Rock_list.get(j);
249         if(Crash(x,y,rock.x,rock.y,h_w,h_h,r_w,r_h)) {
250             //setVisible(false);
251             t1.interrupt();
252             Draw_End();
253             System.out.println("end");
254             t1.notify();
255         }
256     }
257 }
258 }
259 }

```

```
287 public int ImageWidthValue(String file) {  
288     int x = 0;  
289     try {  
290         File f = new File(file);  
291         BufferedImage bi = ImageIO.read(f);  
292         x=bi.getWidth();  
293     }catch(Exception e) {}  
294     return x;  
295 }  
296  
297 public int ImageHeightValue(String file) {  
298     int y =0 ;  
299     try {  
300         File f = new File(file);  
301         BufferedImage bi = ImageIO.read(f);  
302         y=bi.getHeight();  
303     }catch(Exception e) {}  
304     return y;  
305 }  
306  
307 public boolean Crash(int x1, int y1, int x2, int y2, int w1, int h1, int w2, int h2) {  
308     boolean check = false;  
309     if(Math.abs((x1+w1/2)-(x2+w2/2))<(w2/2+w1/2)&&Math.abs((y1+h1/2)-(y2+h2/2))<(h2/2+h1/2)) {  
310         check = true;  
311     }else{check = false;}  
312     return check;  
313 }  
314
```

3

시연

4

보완할 점

Ending 장면이 안들어감

유령과 햄스터의 충돌 여부를 사진 크기로 파악함으로써 발생하는 오차

감사합니다
