Snake 게임의 이해 및 변형

서창환, 신영룡, 조송하, 박현서

# 1. 스네이크 게임 코드 분석 및 플로우차트

|  |  |
| --- | --- |
|  | function setup() {  createCanvas(playfield, 640);  background(51);  s = new Snake(); // s가 뱀이 된다.  frameRate (10);  pickLocation(); // 먹이 생성  } |
| setup() 함수 플로우차트 | setup() 함수 코드 |

|  |  |
| --- | --- |
|  | function draw() {  background(51);  scoreboard();  if (s.eat(food)) {  // 뱀이 food를 먹을 경우, food를 새로 생성  pickLocation();  }  s.death(); // 뱀 죽음여부 확인  s.update(); // 뱀 좌표 업데이트  s.show(); // 뱀 출력  fill (255, 0, 100);  rect(food.x, food.y, scl, scl);  } |
| **draw() 함수 플로우차트** | **draw() 함수 코드** |

|  |  |
| --- | --- |
| 텍스트, 표지판이(가) 표시된 사진  자동 생성된 설명  Y | function pickLocation() {  // food 생성  var cols = floor(playfield/scl);  var rows = floor(playfield/scl);  food = createVector(floor(random(cols)), floor(random(rows)));  food.mult(scl);  // food 벡터의 x, y에 scl 곱하기.  for (var i = 0; i < s.tail.length; i++) {  // 뱀의 몸통이 먹이에 겹치는지 확인  var pos = s.tail[i];  var d = dist(food.x, food.y, pos.x, pos.y);  if (d < 1) {  pickLocation(); // 겹치면 먹이 재생성  }  }  } |
| **pickLocation() 함수 플로우차트** | **pickLocation() 함수 코드** |

|  |  |
| --- | --- |
|  | function scoreboard() {  // 스코어 보드  fill(0);  rect(0, 600, 600, 40); // 검정색 바(스코어 바) 생성  fill(255);  textFont("Georgia"); // 폰트, 사이즈  textSize(18);  text("Score: ", 10, 625); // 점수 텍스트 출력  text("Highscore: ", 450, 625);  text(s.score, 70, 625);  text(s.highscore, 540, 625);  } |
| **scoreboard() 함수 플로우차트** | **scoreboard() 함수 코드** |

|  |  |
| --- | --- |
|  | this.eat = function(pos) {  var d = dist(this.x, this.y, pos.x, pos.y);  // 뱀의 머리와 꼬리 사이 거리  if (d < 1) {  // 뱀이 먹이를 먹으면 점수와 꼬리 각각 +1  this.total++;  this.score++;  text(this.score, 70, 625);  if (this.score > this.highscore) {  // 점수가 최대점수보다 높을 경우  this.highscore = this.score;  // 최대점수 = 현재점수로 갱신  }  text(this.highscore, 540, 625);  return true; // 먹으면 true  } else {  return false; // 안 먹으면 false  }  } |
| **this.eat 플로우차트** | **this.eat 코드** |

|  |  |
| --- | --- |
|  | this.death = function() {  // 뱀이 죽었는지 확인  for (var i = 0; i < this.tail.length; i++) {  var pos = this.tail[i];  var d = dist(this.x, this.y, pos.x, pos.y);  // 머리가 꼬리 또는 벽에 닿을 시, 점수와 꼬리 초기화  if (d < 1) {  this.total = 0;  this.score = 0;  // 초기값은 1이지만 죽으면 0으로 세팅  this.tail = [];  }  }  } |
| **this.death 플로우차트** | **this.death 코드** |

|  |  |
| --- | --- |
|  | this.show = function() {  // 뱀을 출력  fill(255);  for (var i = 0; i < this.tail.length; i++) {  rect(this.tail[i].x, this.tail[i].y, scl, scl); // 꼬리  }  rect(this.x, this.y, scl, scl); // 머리  } |
| **this.show 플로우차트** | **this.show 코드** |

|  |  |
| --- | --- |
|  | this.update = function() {  // 뱀 이동 업데이트  if (this.total === this.tail.length) {  print(this.total, this.tail);  for (var i = 0; i < this.tail.length-1; i++) {  this.tail[i] = this.tail[i+1];  // 배열에 하나 추가  }  // 뱀의 길이와 꼬리 길이가 같을 경우,  // 꼬리 배열 (인덱스벡터)를 한 칸씩 이동  }    this.tail[this.total-1] = createVector(this.x, this.y);  // 머리와 가장 가까운 꼬리 업데이트  this.x = this.x + this.xspeed\*scl;  this.y = this.y + this.yspeed\*scl;  // 이동한 만큼 머리 이동  this.x = constrain(this.x, 0, playfield-scl);  this.y = constrain(this.y, 0, playfield-scl);  // 뱀의 이동 범위 제한  } |
| **this.update 플로우차트** | **this.update 코드** |

# 2. 스네이크 게임 코드의 변형

## 1) food에 이미지를 추가하였습니다.

|  |
| --- |
| function preload() {  img = loadImage("meat.png");  }; |

위의 preload를 통해 이미지를 불러온 후,

|  |  |
| --- | --- |
| function draw() {  background(51);  scoreboard();  if (s.eat(food)) {  pickLocation();  }  s.death();  s.update();  s.show();  fill (255, 0, 100);  rect(food.x, food.y, scl, scl);  } | function draw() {  background(51);  scoreboard();  if (s.eat(food)) {  pickLocation();  }  s.death();  s.update();  s.show();  image(img, food.x, food.y, scl\*1.2, scl\*1.2); // food(meat)  } |
| **기존코드** | **변경된 코드** |

위와 같이 수정하여, 기존의 빨간색 1칸짜리 블록이던 food를 고기 이미지로 대체하였습니다.

|  |  |
| --- | --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |  |
| **변경 전** | **변경 후**  (기존의 붉은색 블록에서 고기 이미지로의 변화) |

.