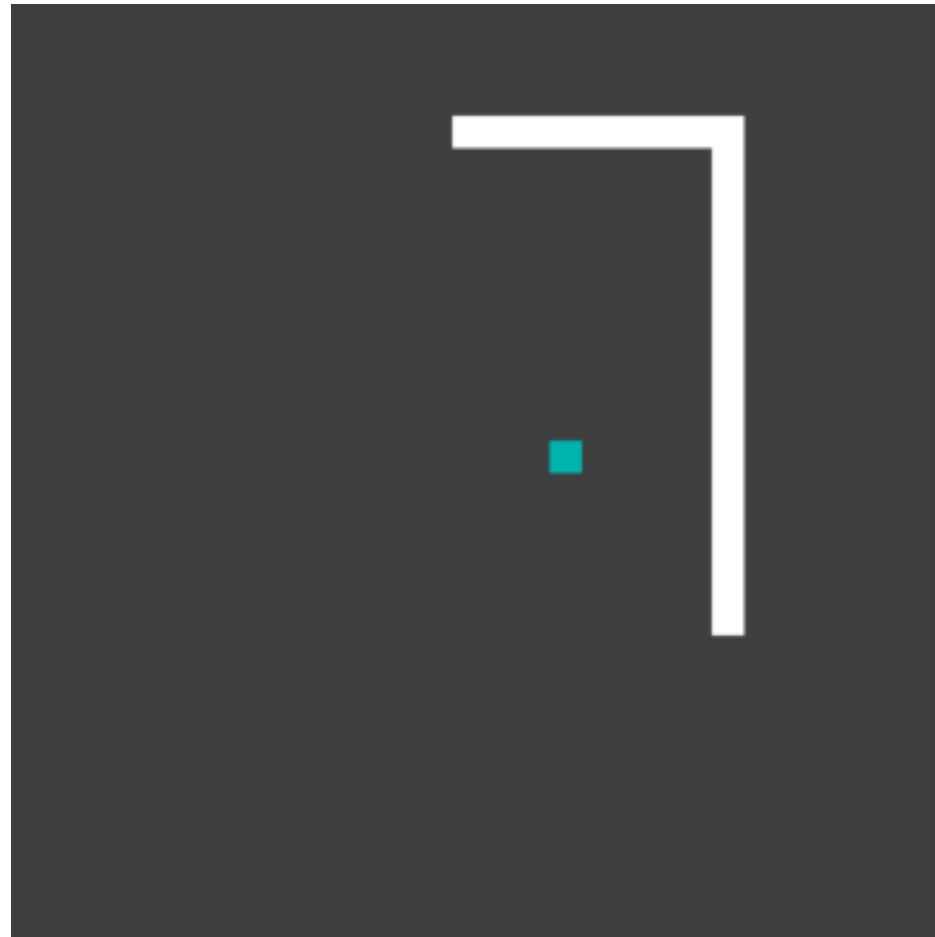


원본게임 소스분석

2014758085 이승찬

SNAKE



목차

- 원본 소스 구성
- 원본 소스 분석
- 개선사항 및 설계

Game Files

index.html

draw.js

화면 출력 및 입력

fruit.js

snake.js

과일 생성

플레이어 로직

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Snake Game</title>
</head>
<body>
  <canvas class="canvas" height="300" width="300"
    style="background-color: #3e3e3e">

  </canvas>
  <h1 class="score"></h1>
</body>
<script type="text/javascript" src="fruit.js"></script>
<script type="text/javascript" src="snake.js"></script>
<script type="text/javascript" src="draw.js"></script>
</html>
```

draw.js

```
const canvas = document.querySelector(".canvas");  
const ctx = canvas.getContext("2d");  
const scale = 10; // 한 칸의 단위  
const rows = canvas.height / scale;  
const columns = canvas.width / scale;  
var snake;
```

...

draw.js

```
(function setup() { // 즉시실행 함수
  snake = new Snake();
  fruit = new Fruit();
  fruit.pickLocation();
                                // 익명함수 정의
  window.setInterval( () => {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    fruit.draw();
    snake.update();
    snake.draw();

    if (snake.eat(fruit)) { fruit.pickLocation(); }
    snake.checkCollision();
    document.querySelector('.score')
      .innerText = snake.total;
  }, 250);
})();
```

draw.js

evt를 받는 익명함수

```
window.addEventListener('keydown', ((evt) => {  
  // 입력된 키에서 "Arrow"를 제거 후 changeDirection의 인수로 넘김  
  const direction = evt.key.replace('Arrow', '');  
  snake.changeDirection(direction);  
})));
```


fruit.js

```
function Fruit() {  
  this.x;  
  this.y;  
  
  this.pickLocation = function() {  
    this.x = (Math.floor(Math.random() *  
      columns - 1) + 1) * scale;  
    this.y = (Math.floor(Math.random() *  
      rows - 1) + 1) * scale;  
  }  
  
  this.draw = function() {  
    ctx.fillStyle = "#4caf50";  
    ctx.fillRect(this.x, this.y, scale, scale)  
  }  
}
```

snake.js

```
function Snake() {  
  this.x = 0;  
  this.y = 0;  
  
  // 플레이어 방향 변수  
  this.xSpeed = scale * 1;  
  this.ySpeed = 0;  
  
  // 먹은 과일 합계  
  this.total = 0;  
  
  // key value를 가질예정  
  this.tail = [];  
  
  ...  
  
  ...  
  
  ...  
}
```

snake.js

```
this.draw = function() {  
  ctx.fillStyle = "#FFFFFFF";  
  // 꼬리 그리기  
  for (let i=0; i<this.tail.length; i++) {  
    ctx.fillRect(this.tail[i].x,this.tail[i].y, scale, scale);  
  }  
  
  // 머리 그리기  
  ctx.fillRect(this.x, this.y, scale, scale);  
}  
  
...
```

snake.js

```
this.update = function() {  
  // this.tail[0]은 제일 뒤쪽,  
  // this.tail[length - 1]은 머리 바로뒤.  
  for (let i=0; i<this.tail.length - 1; i++) {  
    //  
    this.tail[i] = this.tail[i+1];  
  }  
  
  // 머리 바로 뒤 꼬리를 현재 머리위치로  
  this.tail[this.total - 1] = { x: this.x, y: this.y };  
  
  // 머리위치 갱신  
  this.x += this.xSpeed;  
  this.y += this.ySpeed;  
  ...  
}
```

snake.js - update()

```
for (let i=0; i<this.tail.length-1; i++) {  
  this.tail[i] = this.tail[i+1];  
}
```

tail[0] (0,0)	tail[1] (0,1)	tail[2] (0,2)	tail[3] (1,2)	HEAD (2,2)
------------------	------------------	------------------	------------------	---------------

0,0	0,1	0,2	
		1,2	
		2,2	

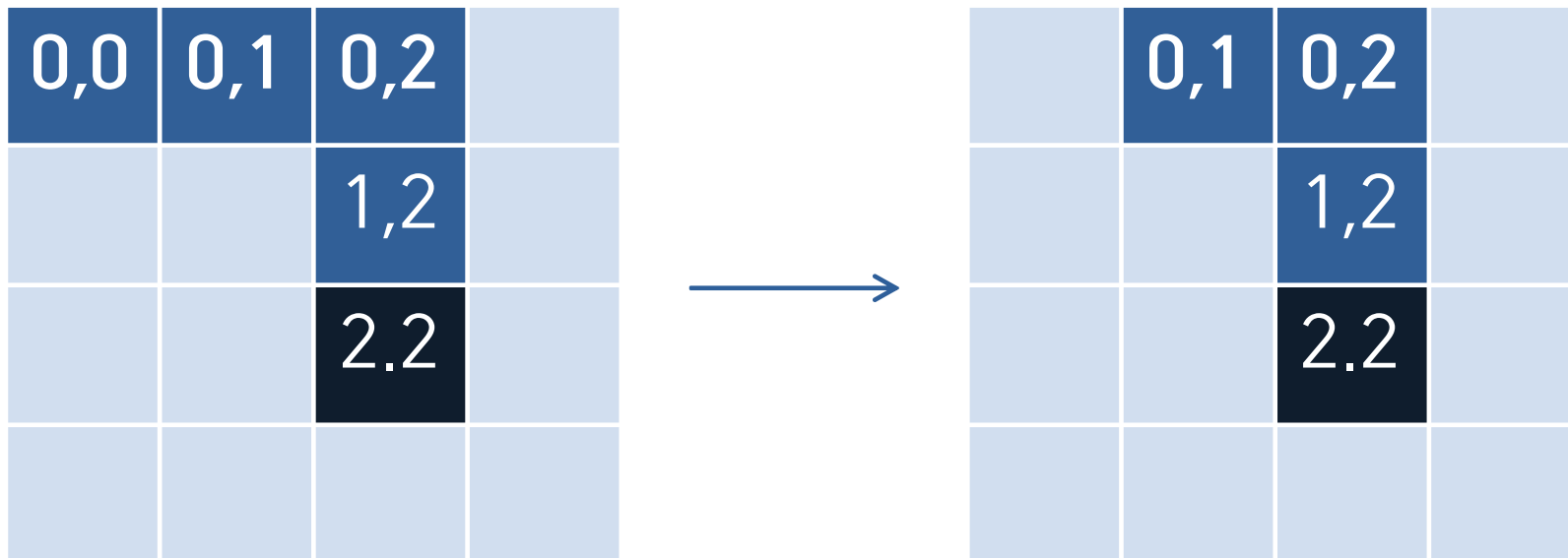


tail[]과 HEAD는
같은 배열이 아님

snake.js - update()

```
for (let i=0; i<this.tail.length-1; i++) {  
  this.tail[i] = this.tail[i+1];  
}
```

tail[0] (0,1)	tail[1] (0,2)	tail[2] (1,2)	tail[3] (1,2)	HEAD (2,2)
------------------	------------------	------------------	------------------	---------------

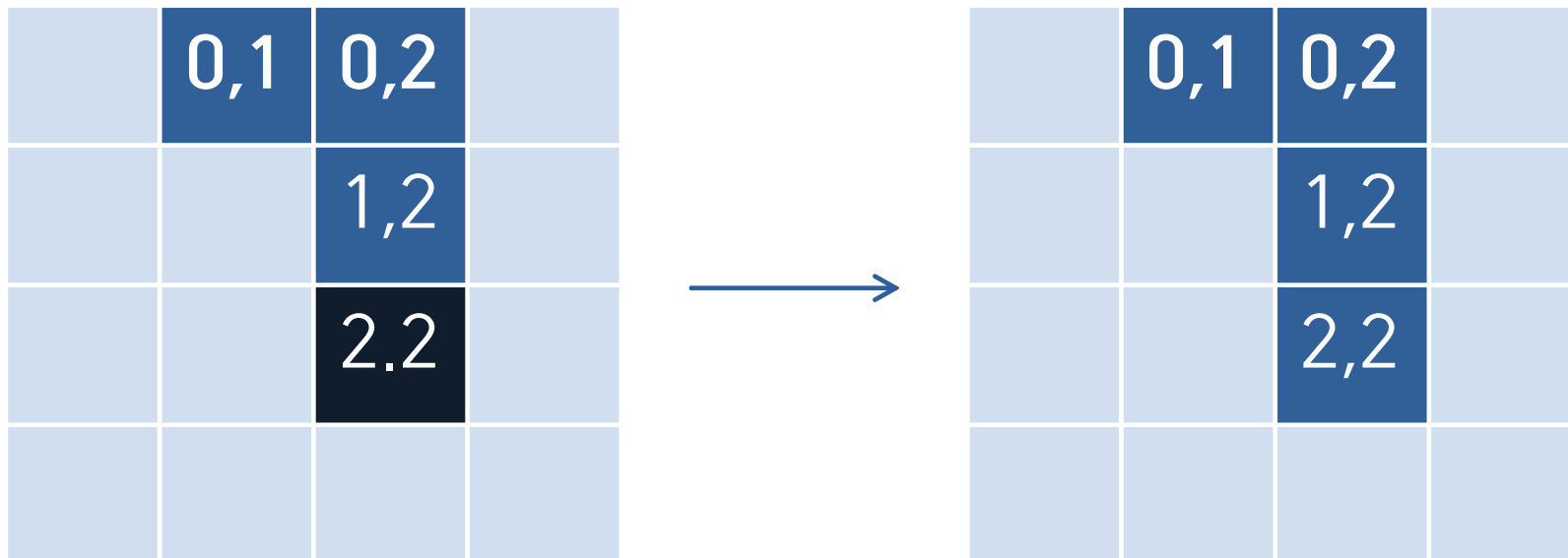


snake.js - update()

// 머리 바로 뒤 꼬리를 현재 머리위치로

```
this.tail[this.total - 1] = { x: this.x, y: this.y };
```

tail[0] (0,1)	tail[1] (0,2)	tail[2] (1,2)	tail[3] (2,2)	HEAD (2,2)
------------------	------------------	------------------	------------------	---------------

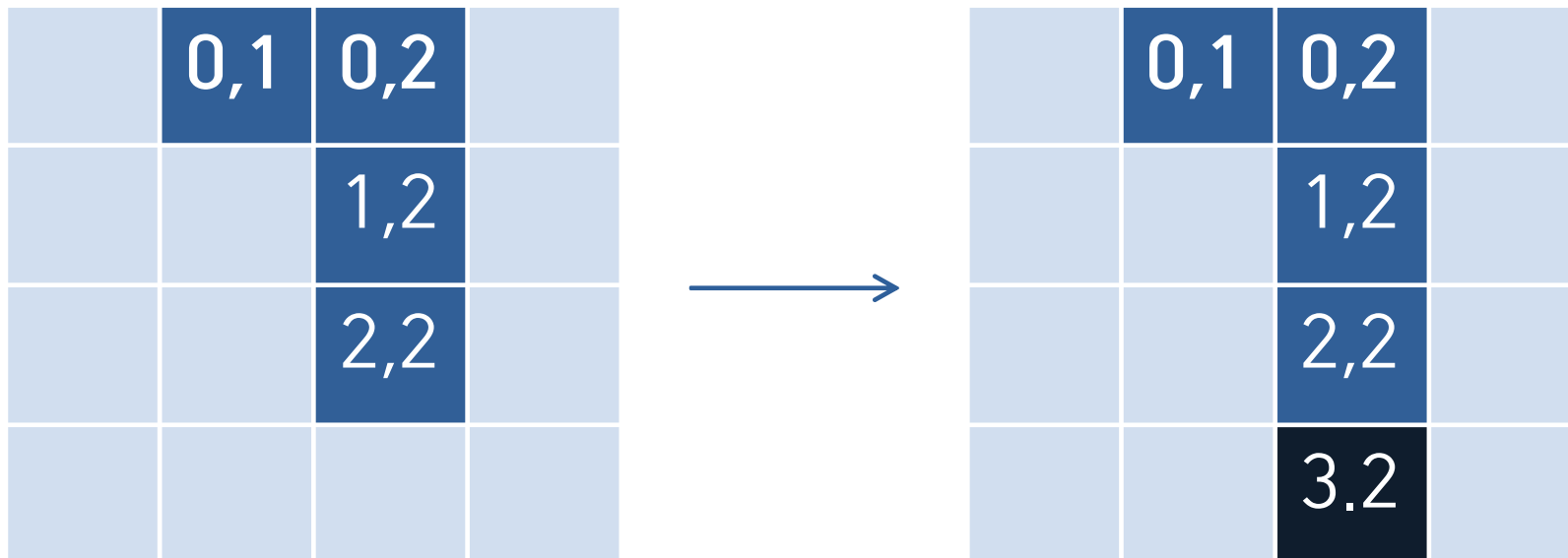


snake.js - update()

```
this.x += this.xSpeed;
```

```
this.y += this.ySpeed;
```

tail[0] (0,1)	tail[1] (0,2)	tail[2] (1,2)	tail[3] (2,2)	HEAD (3,2)
------------------	------------------	------------------	------------------	---------------



snake.js

```
// 화면 끝으로 갈 시 반대쪽으로 나옴
  if (this.x > canvas.width) {
    this.x = 0;
  }
  if (this.y > canvas.height) {
    this.y = 0;
  }
  if (this.x < 0) {
    this.x = canvas.width;
  }
  if (this.y < 0) {
    this.y = canvas.height;
  }
}
...
```

snake.js

```
this.changeDirection = function(direction) {  
  switch(direction) {  
    case 'Up':  
      this.xSpeed = 0;    this.ySpeed = -scale * 1;    break;  
    case 'Down':  
      this.xSpeed = 0;    this.ySpeed = scale * 1;    break;  
    case 'Left':  
      this.xSpeed = -scale * 1;    this.ySpeed = 0;    break;  
    case 'Right':  
      this.xSpeed = scale * 1;    this.ySpeed = 0;    break;  
  }  
}  
...  
...
```

snake.js

```
this.eat = function(fruit) {  
  if (this.x === fruit.x && this.y === fruit.y) {  
    this.total++;  
    return true;  
  }  
  
  return false;  
}  
...
```

snake.js

```
this.checkCollision = function() {  
  for (var i=0; i<this.tail.length; i++) {  
    // 플레이어 몸에 닿으면  
    if (this.x === this.tail[i].x &&  
        this.y === this.tail[i].y) {  
  
      // 초기화  
      this.total = 0;  
      this.tail = [];  
    }  
  }  
}
```

게임 기획

개선 및 추가, 변경

개선 및 추가, 변경

기타

- * 캡슐화 getter setter
- * 인게임 현재 스코어 추가
- * 모드별 하이스코어 표시
- * 타이틀 화면 추가
- * BGM, SFX 추가
- * Map 밖으로 -> 게임오버로 변경

UI

- * Window 크기 기반 Canvas 중앙배치
- * Map 캔버스 디자인 개선
- * 타일 기반 Map으로 원하는 Map 크기 선택 가능
- * 결과화면 추가
 - + 결과화면 fade in 추가

개선 및 추가, 변경

플레이어

- * 플레이어 색상 변경
- * 꼬리 그라데이션 추가

로직

- * 진행방향과 반대되는 input 무시
- * 기존 pickLocation 함수 개선
- * 키 입력 컨트롤러 버퍼추가
- * 기존 draw함수 최적화

개선 및 추가, 변경

노멀모드

- 일정 시간동안 먹이를 먹지 않을시
게임오버 + 카운터
- 일정한 배수의 과일을 먹으면
이동속도 증가
- 일정 확률로 꼬리 줄이는
아이템 스폰

하드모드

- 하드모드 선택시 Map 색상 바뀜
- 먹이를 먹으면 꼬리 끝에 흔적을 남김
접촉하면 게임오버
- 일정 확률의 꼬리 줄이는 아이템 스폰
일정 시간지나면 사라짐

감사합니다.