Mad Scientist

김기문 김민준 김유빈 한명지

발표 개요

1 컨셉

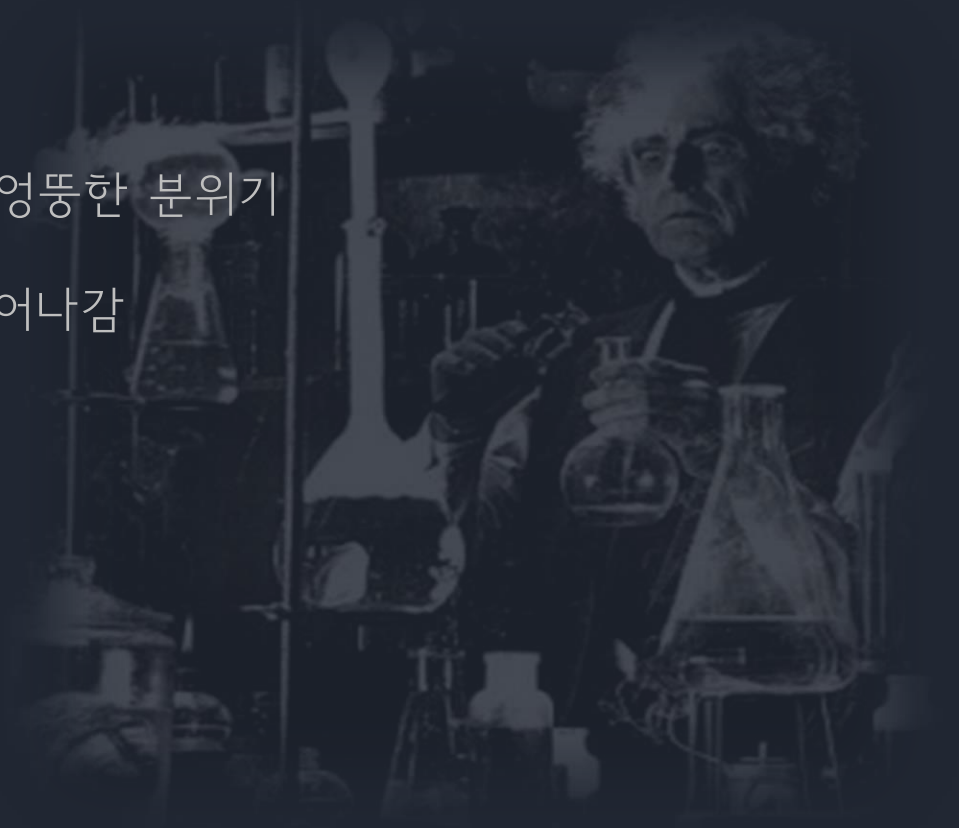
2 게임의 재미 요소

3 게임의 진행과 구현 방법

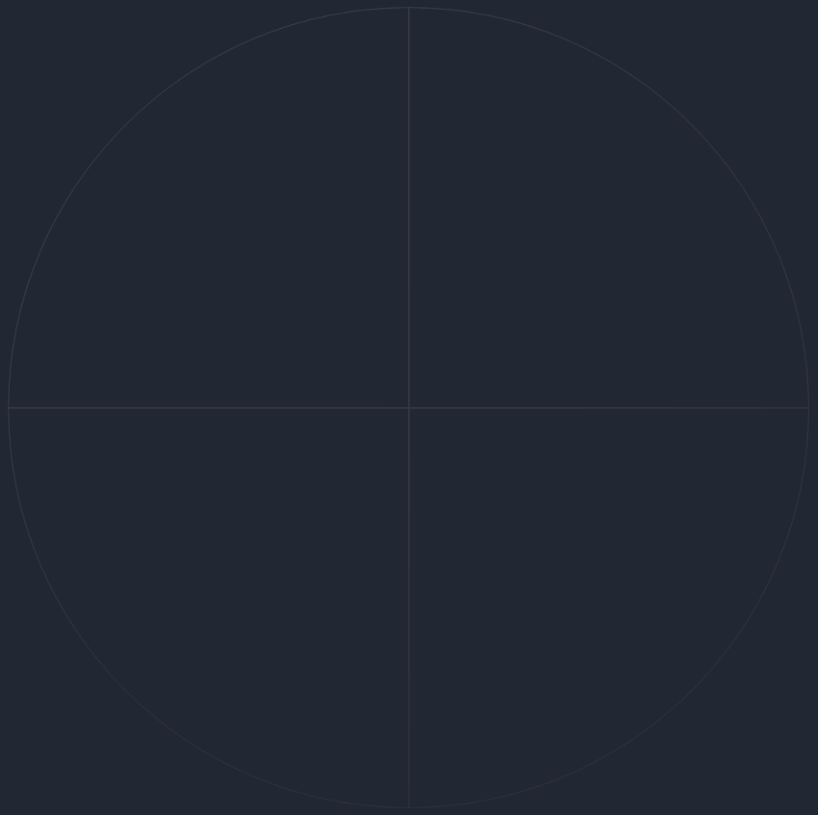
4 협업 과정

방 탈출 컨셉

- 미친 과학자의 방에서 탈출하는 내용
- 마치 과학자의 방 같은 컨셉
- 다소 괴상하고 공포스럽거나, 가끔은 엉뚱한 분위기
- 과학 실험같은 힌트를 통해 단서를 풀어나감



게임의 재미 요소
3가지



다양한 사운드

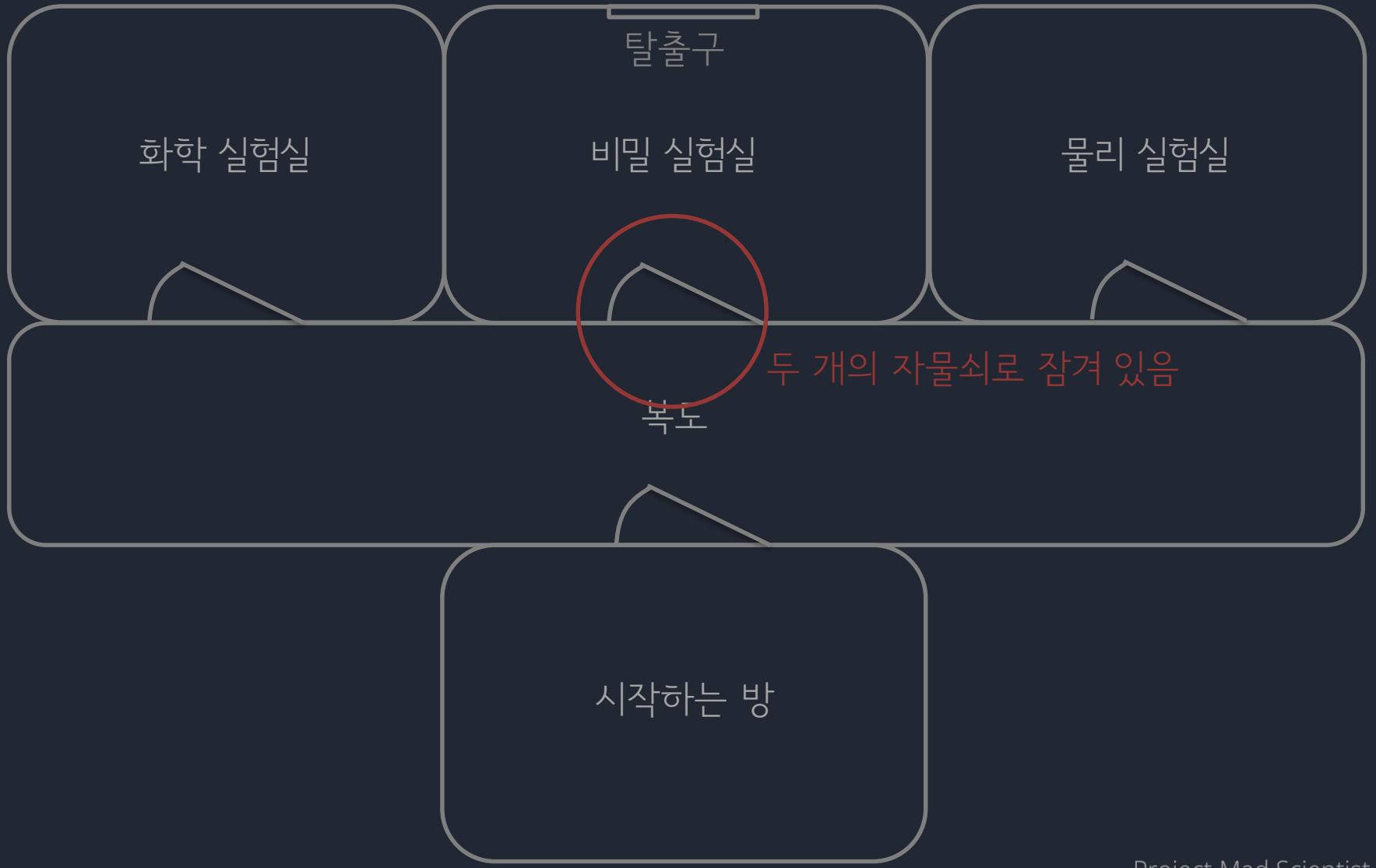
미치광이 과학자라는 설정의 게임 구성

반복적이지 않은 다양한 문제 구성

게임 진행

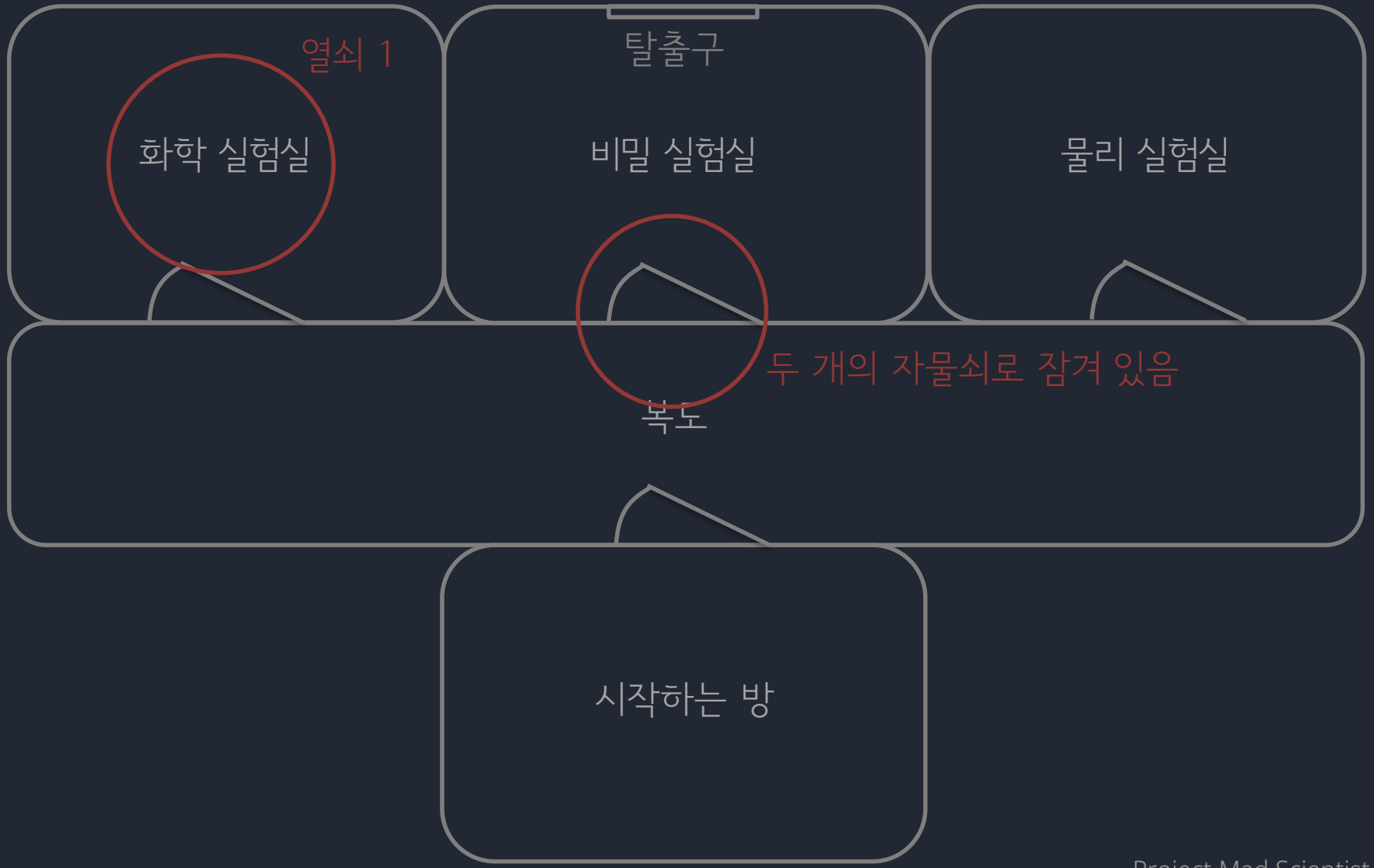
게임 진행

Map



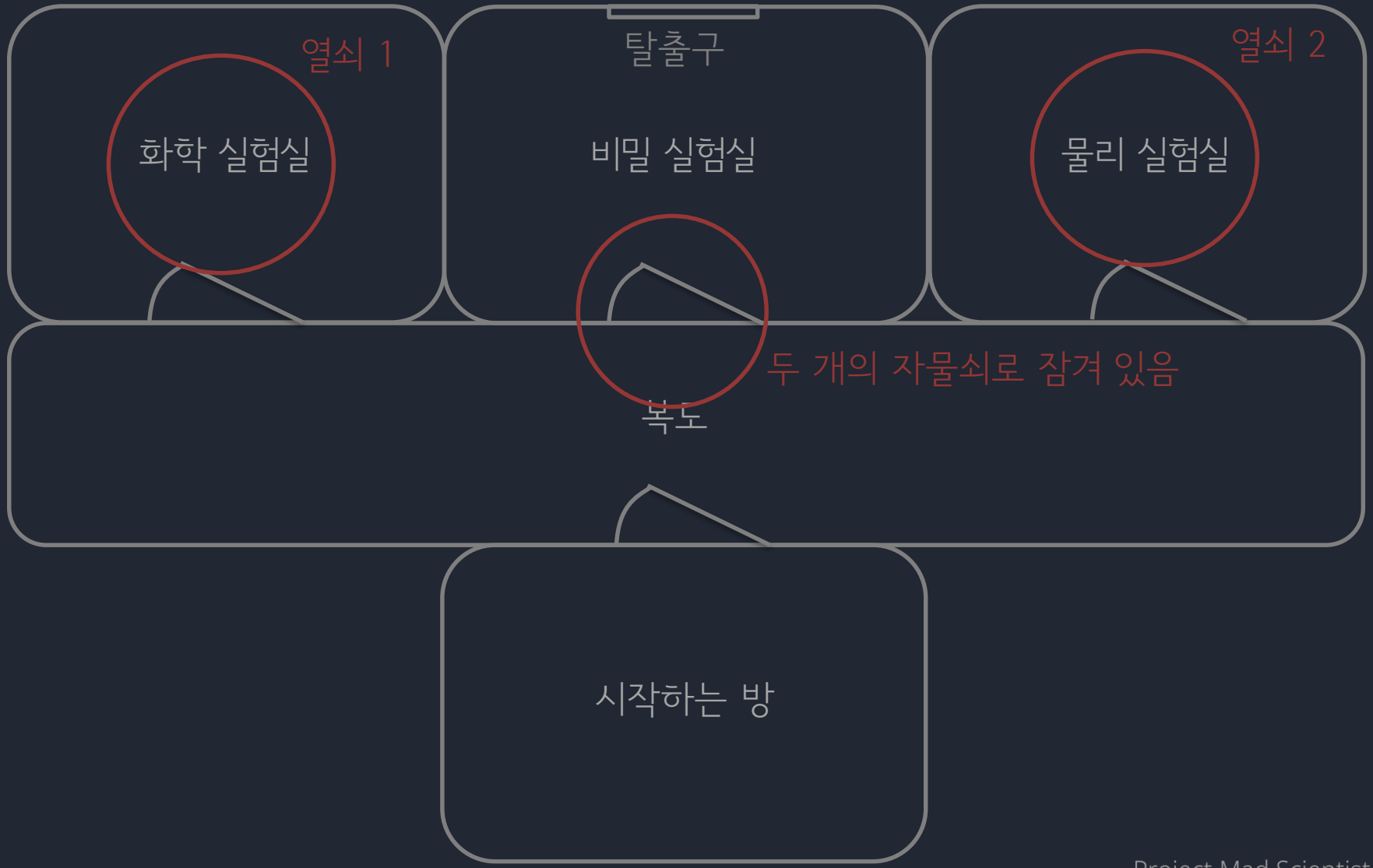
게임 진행

Map



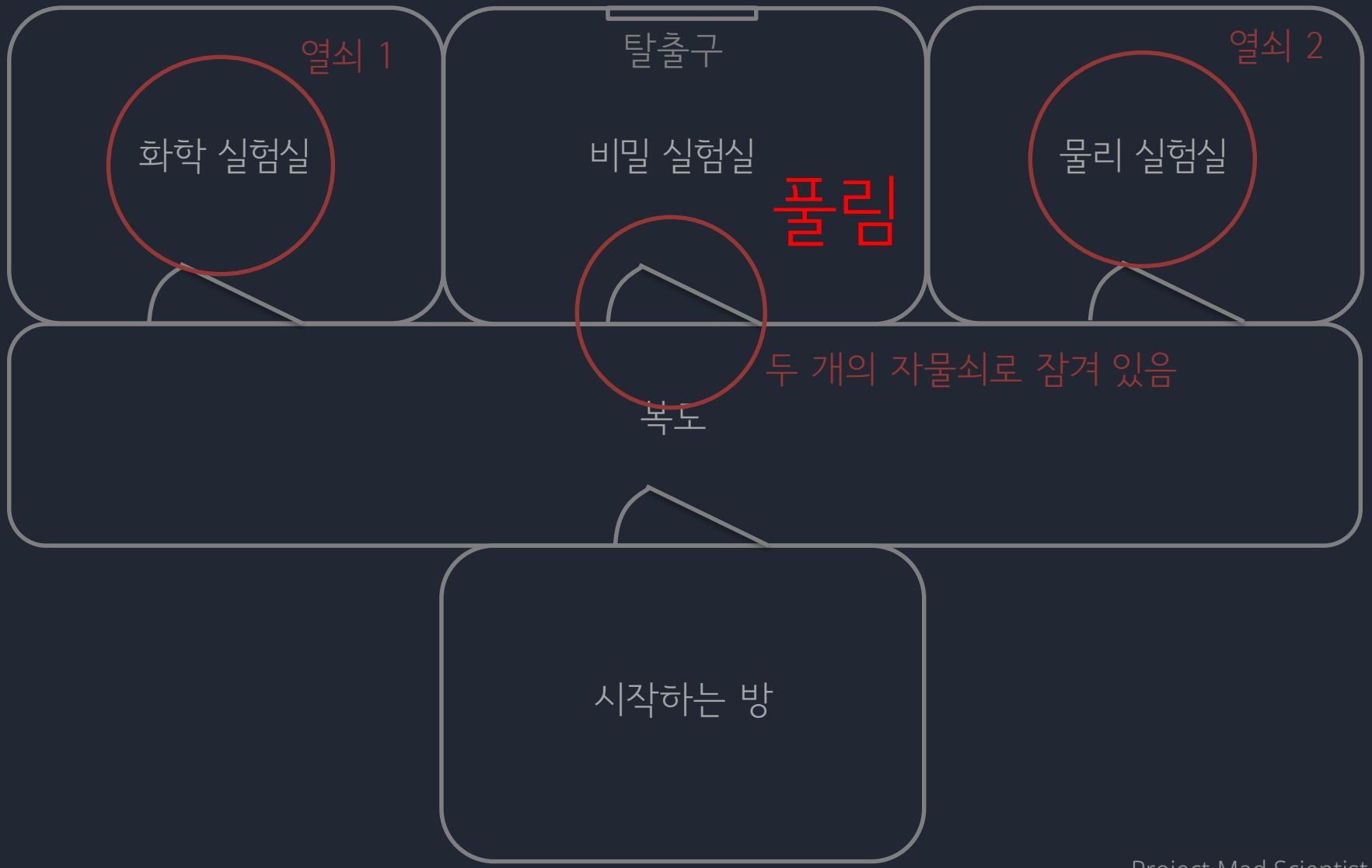
게임 진행

Map

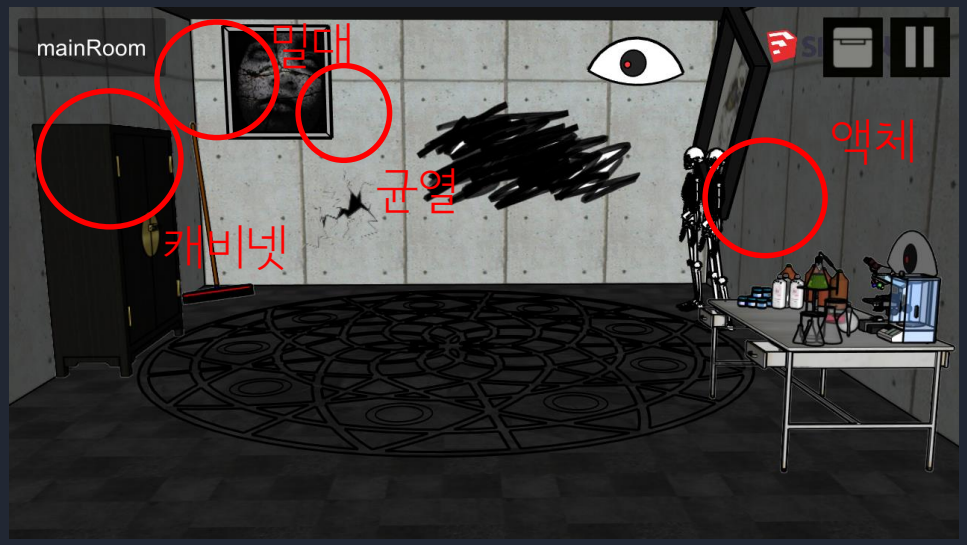


게임 진행

Map



게임 진행

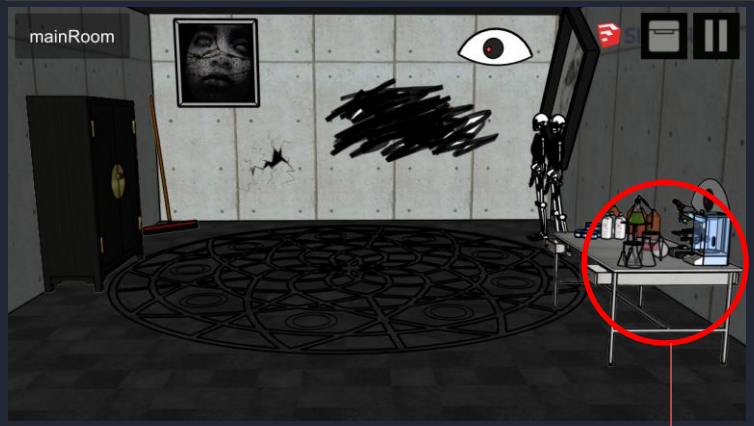


" 시작하는 방 "

- 밀대를 분해하여 막대기를 얻는다.
- 실험 책상 위에서 과학자의 일지를 읽어 액체에 대한 힌트를 얻는다.
- 정체불명의 액체를 얻는다.
- 액체를 벽의 낙서에 뿌려 캐비닛 비밀번호를 확인한다.
- 캐비닛을 열고 망치 머리를 얻는다.
- 막대기와 망치 머리를 조립해 망치로 벽의 균열을 부수고 나간다.

게임 진행

click to move



onClick()

onClick()

새로운 방(room) 객체를 만들어
onClick() 함수에 moveto() 삽입

getItem()

#Definition

```
function getItem(room, name, image,sound){
    Object.call(this, room, name, image)

    this.sound=sound
}

getItem.prototype = new Object()   // inherited from Object

//item의 onClick - 줍기
getItem.member('onClick', function(){
    playSound(this.sound)
    this.id.pick()
})
```

getItem()

#usage_example

```
//chemistry

chemistry.beaker1 = new getItem(chemistry, 'beaker1', 'beaker1.png','glass.wav')
chemistry.beaker1.resize(40)
chemistry.beaker1.locate(800,295)


chemistry.beaker2 = new getItem(chemistry, 'beaker2', 'beaker2.png','glass.wav')
chemistry.beaker2.resize(45)
chemistry.beaker2.locate(1150,420)
```

게임 진행

"화학 실험실 "

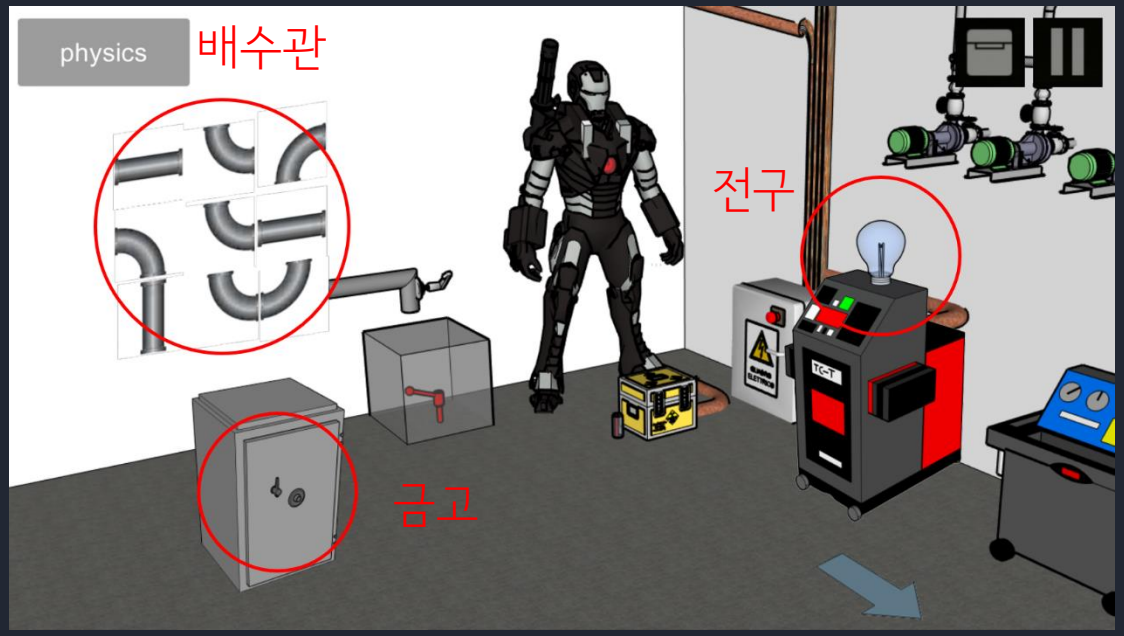
- 화학물질이 묻어진 종이를 주워 특수용액으로 씻어 금고 비밀번호를 얻는다.
- 금고를 열어 열쇠주형과 열쇠를 만드는 방법의 연구일지를 얻는다.
- 연구일지의 내용처럼 책상에서 얻을 수 있는 두용액을 혼합 한 후 주형을 이용해 모양을 맞추고 이를 분리한다.
- 혼합액으로 만든 열쇠모양을 UV 기계를 이용해 굳혀 비밀실험실방으로 통하는 열쇠를 얻는다.

Chemistry Room

```
chemistry_uv.uv.onClick=function(){
    if(chemistry.key2.isHanded()&&chemistry_uv.uv.isClosed()&&chemistry_uv.key2.isLocked()){
        chemistry_uv.uv.open()
        chemistry_uv.key2.show()
    }
}

chemistry_uv.switch.onClick=function(){
    if(chemistry_uv.switch.isLocked()&&chemistry_uv.uv.isClosed()){
        playSound("switch.wav")
        chemistry_uv.switch.open()
        chemistry_uv.uv.open()

    }else if(chemistry_uv.switch.isOpened()&&chemistry_uv.uv.isOpened()){
        playSound("switch.wav")
        chemistry_uv.switch.lock()
        chemistry_uv.uv.close()

    }else if(chemistry_uv.switch.isLocked()&&chemistry_uv.uv.isOpened()&&chemistry_uv.key2.isLocked()){
        playSound("switch.wav")
        chemistry_uv.switch.open()
        chemistry_uv.uv.close()
        chemistry_uv.key2.hide()
    }else if(chemistry_uv.switch.isOpened()&&chemistry_uv.uv.isClosed()){
        printMessage('혼합액을 이용한 열쇠가 단단히 굳었다.')
        playSound("switch.wav")
        chemistry_uv.switch.lock()
        chemistry_uv.key.show()
        chemistry_uv.key2.open()
    }
}

chemistry_uv.switch.onOpen=function(){
    chemistry_uv.uv.setSprite("uvclose.png")
    chemistry_uv.switch.setSprite("switchon.png")
}

chemistry_uv.switch.onLock=function(){
    chemistry_uv.uv.setSprite("uvopen.png")
    chemistry_uv.switch.setSprite("switchoff.png")
}
```

"특별한 함수 추가 없이, 이미 생성된 함수를 적절히 조합 및 응용하여 필요 구성요소 구현"

게임 진행



- 배수관 퍼즐을 풀어 상자 속 레버를 얻는다.
- 레버를 전구 기계에 꽂아서 전구에서 힌트를 얻는다.
- 전구에서 얻은 힌트로 금고를 풀어 시크릿 방 열쇠를 얻는다.

“물리 실험실 "

Drain1()

#Definition

```
//---------------------------------Drain1-------------------------------
//이미지 4개가 필요한 배수관
function Drain1(room, name, image1, image2, image3, image4, answer){
    Object.call(this, room, name, image1)

    // Drain1 properties
    this.image1 = image1
    this.image2 = image2
    this.image3 = image3
    this.image4 = image4
    this.count = 0
    this.answer = answer
    this.clear = 0  //정답 맞추면 1, 못 맞추면 0
}
Drain1.prototype = new Object()   // inherited from Object
```

- image1 ~ image4 : 배수관 이미지
- count : 클릭 횟수
- answer : 정답 클릭 횟수
- clear : count == answer인 경우 1 , 그렇지 아니면 0

Drain1()

#Definition

```
Drain1.member('onClick', function(){
    //클릭 횟수 카운트
    if(this.count <= 3){
        this.count += 1
    }
    else if(this.count == 4){
        this.count = 0   //리셋
    }
    //클릭 횟수마다 이미지 변화
    if(this.count == 0){
        this.setSprite(this.image1)
    }
    else if(this.count == 1){
        this.setSprite(this.image2)
    }
    else if(this.count == 2){
        this.setSprite(this.image3)
    }
    else if(this.count == 3){
        this.setSprite(this.image4)
    }

    //특정 클릭 수 -> 한 개 성공
    if(this.answer != undefined){
        if(this.count == this.answer){
            this.clear = 1
        }
        else if(this.count != this.answer) {
            this.clear = 0
        }
    }
})
```

1. count를 0, 1, 2, 3으로 제한한다. 따라서 총 4번 이미지가 변화한다.

2. 4가지의 count에 따라 이미지가 변하도록 한다.

3. clear변수를 지정하는 것으로, 클릭 횟수와 정답 클릭 횟수가 같을 때를 알려준다.

Drain1()

#Definition

```
//게임 리셋 함수
Drain1.member('Reset', function(){
    this.setSprite(this.image1) //이미지 원상복귀
    this.count = 0
    this.clear = 0
})
```

- 첫 이미지로 변화
- 사용자 클릭 횟수 0으로 초기화
- 정답을 맞췄는지 알려주는 clear도 0으로 초기화

Drain1()

#Definition

```
drain_close.velve.onClick = function(){
    if(drain_close.velve.isLocked()){
        playSound('bump.wav')
        drain_close.velve.open()

        if(drain_close.drain1.clear == 1 && drain_close.drain2.clear == 1 && drain_close.drain3.clear == 1 && dr
            playSound('drain.wav')  //물소리
            drain_close.box.setSprite('물상자.png')
            drain_close.lever.show()
        }
        else{
            drain_close.drain1.Reset()
            drain_close.drain2.Reset()
            drain_close.drain3.Reset()
            drain_close.drain4.Reset()
            drain_close.drain5.Reset()
            drain_close.drain6.Reset()
            drain_close.drain7.Reset()
            drain_close.drain8.Reset()
            drain_close.drain9.Reset()

        }
    }
    else if(drain_close.velve.isOpened()){
        playSound('locked.wav')
        drain_close.velve.lock()
    }
}
```

Drain2()

#Definition

```
//------------------------------Drain2-----------------------
//이미지 2개가 필요한 배수관(직선 배수관)
function Drain2(room, name, image1, image2, answer){
    Object.call(this, room, name, image1)

    // Drain2 properties
    this.image1 = image1
    this.image2 = image2
    this.count = 0
    this.answer = answer
    this.clear = 0  //정답 맞추면 1, 못 맞추면 0
}
Drain2.prototype = new Object()   // inherited from Object
```

- 이미지 개수가 2개인 것을 제외하고 Drain1과 동일

Bulb()

#Definition

```
//----------------------------Bulb----------------------------
function Bulb(room, name, image0, image1, image2, image3, image4, num){
    Object.call(this, room, name, image0)

    // Bulb properties
    this.image0 = image0
    this.image1 = image1
    this.image2 = image2
    this.image3 = image3
    this.image4 = image4

    this.num = num  //이미지 갯수
    this.count = 0  //클릭 횟수 카운트
}
Bulb.prototype = new Object()   // inherited from Object
```

- image0 ~ image4 : 5가지의 이미지(초기화 이미지 포함)
- num : 변하는 이미지 4개
- count : 클릭 횟수

Bulb()

#Definition

```
Bulb.member('Change', function(){
    if(this.count == 0){
        this.setSprite(this.image1)
    }
    else if(this.count == 1){
        this.setSprite(this.image2)
    }
    else if(this.count == 2){
        this.setSprite(this.image3)
    }
    else if(this.count == 3){
        this.setSprite(this.image4)
    }

    if(this.count < this.num){
        this.count++
    }
    else if(this.count >= this.num){
        this.count = 0   //리셋
    }
})
//리셋 함수
Bulb.member('Reset', function(){
    this.count = 0
    this.setSprite(this.image0)
})
```

Safe()

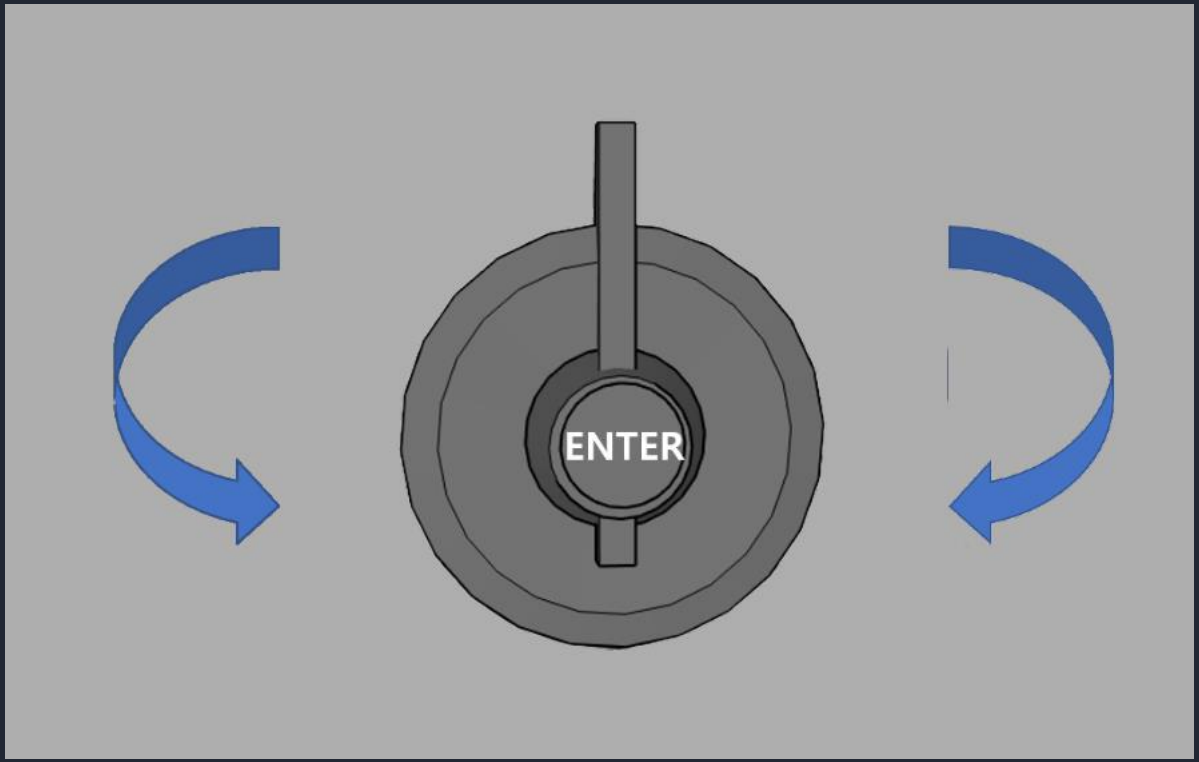
#Definition

```
//------------------------------Safe--------------------------
function Safe(room, name, image0, image1, image2, image3, answer_array){
    Object.call(this, room, name, image0)

    // Safe properties
    this.image0 = image0
    this.image1 = image1
    this.image2 = image2
    this.image3 = image3

    this.current = 0     //다이얼 현재 방향 (0,1,2,3)

    this.safe_array = new Array()  //사용자의 클릭 배열
    this.answer_array = answer_array   //정답 배열
}
Safe.prototype = new Object()   // inherited from Object
```

- image0 ~ image3 : 4가지 이미지
- answer_array : 다이얼 정답 배열
- current : 다이얼 현재 방향(위=0, 오른=1, 아래=2, 왼=3)
- safe_array : 사용자의 다이얼 배열

Safe()

#usage_example



Safe()

#Definition

```
//화살표에 클릭에 따라 이미지 변화 + 사용자 클릭 배열에 저장 함수
Safe.member('Change', function(arrow){  //매개변수는 화살표 방향 (-1 == 왼쪽, 1 == 오른쪽)
    if(arrow == 1){ //arrow = right
        //배열에 추가
        this.safe_array.push(1)

        //image change
        if(this.current == 0){
            this.setSprite(this.image1)
            this.current = 1
        }
        else if(this.current == 1){
            this.setSprite(this.image2)
            this.current = 2
        }
        else if(this.current == 2){
            this.setSprite(this.image3)
            this.current = 3
        }
        else if(this.current == 3){
            this.setSprite(this.image0)
            this.current = 0
        }
    }
```

<오른쪽 화살표>

1. 사용자 배열에 '1' 추가

2. current(현재 다이얼 방향)에 따라 다이얼 이미지 변화 + current 값 변화

Safe()

#Definition

```
    else if(arrow == -1){ //arrow == left
        //배열에 추가
        this.safe_array.push(-1)

        //image change
        if(this.current == 0){
            this.setSprite(this.image3)
            this.current = 3
        }
        else if(this.current == 1){
            this.setSprite(this.image0)
            this.current = 0
        }
        else if(this.current == 2){
            this.setSprite(this.image1)
            this.current = 1
        }
        else if(this.current == 3){
            this.setSprite(this.image2)
            this.current = 2
        }
    }
})
```

<왼쪽 화살표>

1. 사용자 배열에 '-1' 추가

2. current(현재 다이얼 방향)에 따라 다이얼 이미지 변화 + current 값 변화
   -> 오른쪽 화살표와 다른 순서

Safe()

#Definition

```
//---정답 배열과 사용자 배열 비교 함수-----
Safe.member('Compare', function(array1, array2){
    var i, isA1, isA2
    isA1 = Array.isArray(array1)
    isA2 = Array.isArray(array2)   //배열인지 확인

    if(isA1 !== isA2) {    // 매개변수 하나가 배열이 아닌 경우
        return false
    }
    if (! (isA1 && isA2)) {      // 둘 다 배열이 아닌 경우
        return array1 === array2
    }

    if (array1.length != array2.length) { // 배열 길이가 다르면
        return false;
    }

    //각 요소 비교
    for(i = 0; i < array1.length; i ++){
        if(array1[i] != array2[i]){
            return false
        }
    }

    return true
})
```

1. 배열

2. 비교

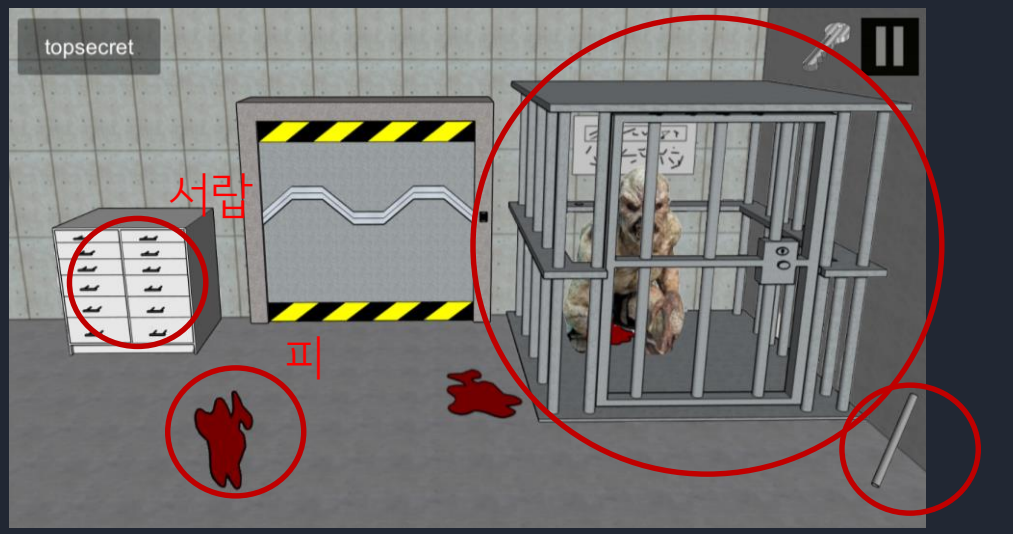
Safe()

#Definition

```
//리셋 함수
Safe.member('Reset', function(){
    this.current = 0
    this.safe_array = []    //사용자 배열 초기화

    this.setSprite(this.image0)
})
```

- current를 0(위쪽)으로 초기화
- safe_array(사용자 입력 배열) 초기화
- 이미지 초기화

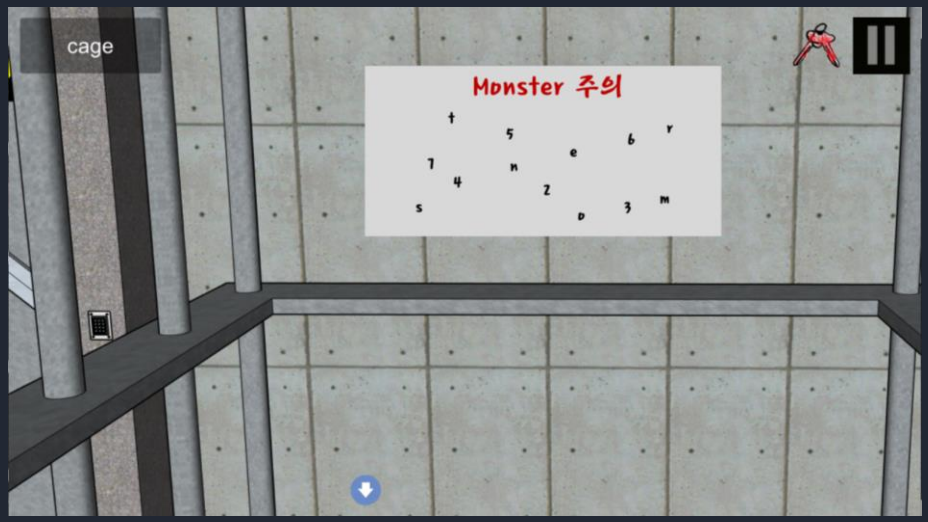
게임 진행

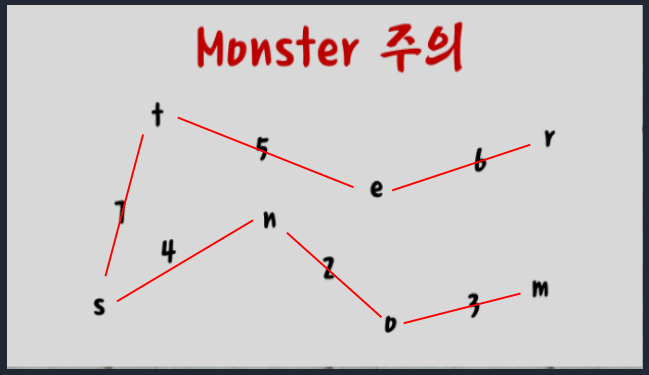


“비밀 실험실 "

- 비밀실험실에선 몬스터가 존재 몬스터를 클릭하면 소리를 지르니 주의
- 서랍을 열어 주사기와 마취제를 얻는 후 이를 조합한다.
- 파이프를 주워 마취제를 주입한 주사기와 합쳐 마취파이프를 제작하여 몬스터를 잠재운다.
- 피를 클릭하여 피 속에 숨겨진 열쇠를 얻은 후 철창을 연다.
- 철장안으로 들어가 퀴즈를 풀어 비밀실험실을 탈출한다.

Quiz





- “Monster 주의” 힌트를 이용해 풀이
- Monster의 영문자를 따라가면 궤적에 숫자가 방탈출의 비밀번호

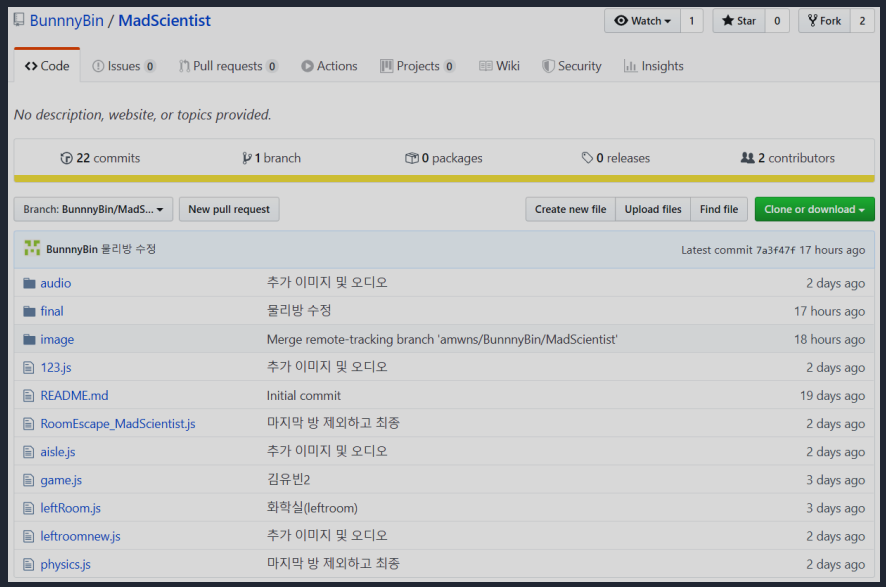
Top Secret Room

```
//감옥
topsecret.cage = new Door(topsecret, 'cage', 'cageclose.png','cageopen.png',cage)
topsecret.cage.resize(600)
topsecret.cage.locate(965,335)
topsecret.cage.lock()

topsecret.blood = new Object(topsecret,'blood','blood.png')
topsecret.blood.locate(300,600)
topsecret.blood.resize(200)
topsecret.blood.lock()

topsecret.blood.onClick=function(){
    if (topsecret.blood.isLocked()){
        printMessage('피속에서 열쇠 하나를 얻었다.')
        topsecret.blood.open()
        topsecret.cagekey.pick()
        playSound('key.wav')
    }
}

topsecret.cagekey = new getItem(topsecret, 'cagekey', 'cagekey.png','key.wav')
topsecret.cagekey.hide()

topsecret.cage.onClick = function(){
    if(topsecret.tranquilizer.isHanded()&&topsecret.monster.isLocked()&&topsecret.cage.isLocked()){
        playSound('monster_dead_sound.wav')
        topsecret.monster.setSprite('dead_monster.png')
        topsecret.cage.setSprite('cageclose2.png')
        topsecret.monster.open()
        printMessage('몬스터를 해치웠다!')
    }else if(topsecret.tranquilizer.isHanded()&&topsecret.monster.isOpened()&&topsecret.cage.isLocked()){
        printMessage('이제 문을 열수 있을거 같아')
    }
    else if(topsecret.cagekey.isHanded()&&topsecret.monster.isOpened()&&topsecret.cage.isLocked()){
        topsecret.cage.open()
        playSound('prison_open.wav')
    }
    else if(topsecret.monster.isOpened()&&topsecret.cage.isOpened()){
        Game.move(cage)
    }
    else{
        printMessage('뒤에 무언가 적혀있는거 같은데 몬스터 때문에 다가가지를 못하겠다... 파이프를 이용해 몬스터를 해치워야겠다')
        playSound('monsteractive.wav')
    }
}
```

"특별한 함수 추가 없이, 이미 생성된 함수를 적절히 조합 및 응용하여 필요 구성요소 구현"

협업 과정

- GitHub Workflow



협업 과정

- GitHub Workflow

BunnnyBin / MadScientist

Watch 1 | Star 0 | Fork 2

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki | Security | Insights

No description, website, or topics provided.

22 commits | 1 branch | 0 packages | 0 releases | 2 contributors

Branch: BunnnyBin/MadS... | New pull request | Create new file | Upload files | Find file | Clone or download

BunnnyBin 물리방 수정 — Latest commit 7a3f47f 17 hours ago

audio	추가 이미지 및 오디오	2 days ago
final	물리방 수정	17 hours ago
image	Merge remote-tracking branch 'amwns/BunnnyBin/MadScientist'	18 hours ago
123.js	추가 이미지 및 오디오	2 days ago
README.md	Initial commit	19 days ago
RoomEscape_MadScientist.js	마지막 방 제외하고 최종	2 days ago
aisle.js	추가 이미지 및 오디오	2 days ago
game.js	김유빈2	3 days ago
leftRoom.js	화학실(leftroom)	3 days ago
leftroomnew.js	추가 이미지 및 오디오	2 days ago
physics.js	마지막 방 제외하고 최종	2 days ago

- Offline Meeting

Q&A

Principles & Methods to go Beyond UX