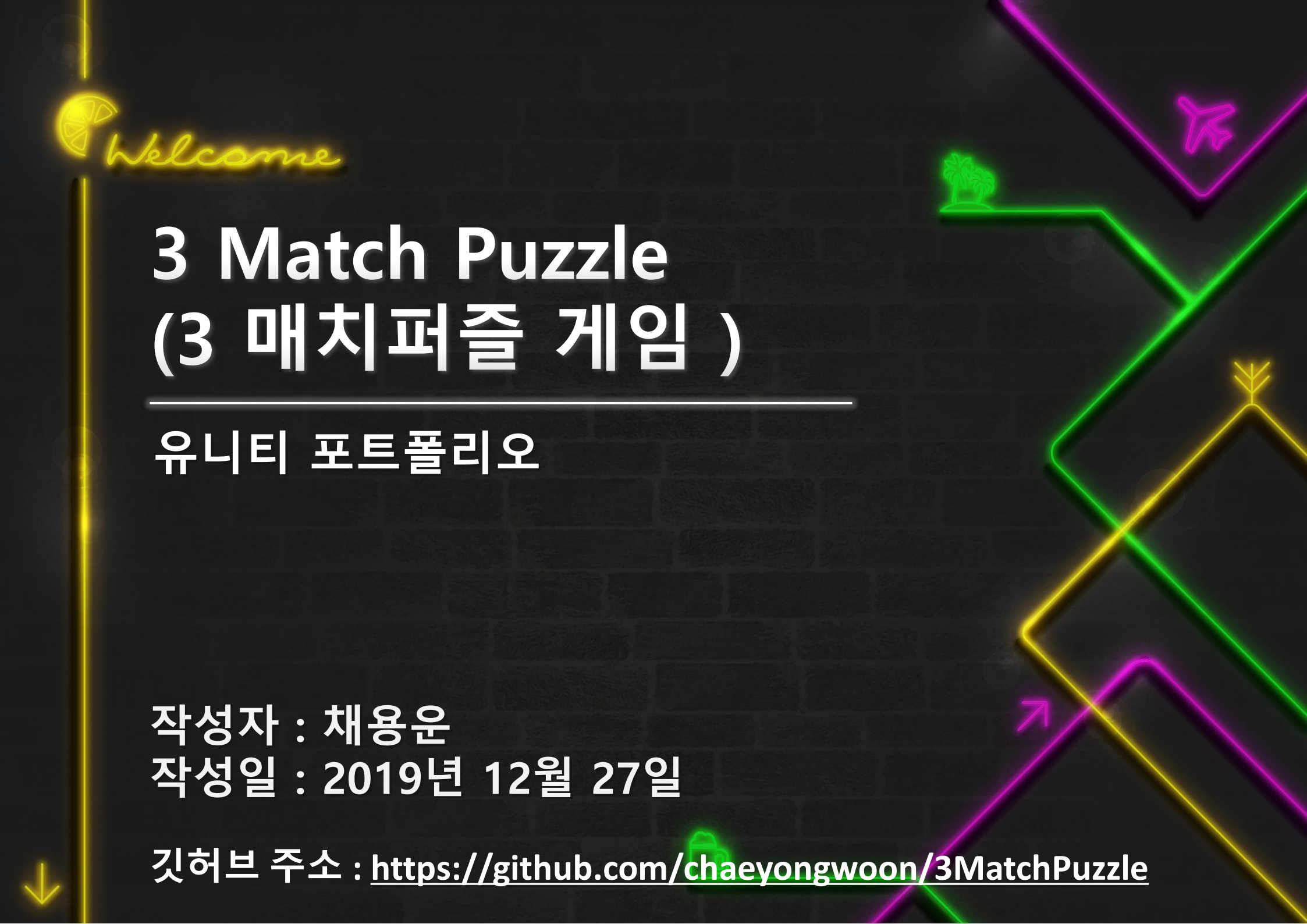


3 Match Puzzle
(3 매치퍼즐 게임)

유니티 포트폴리오

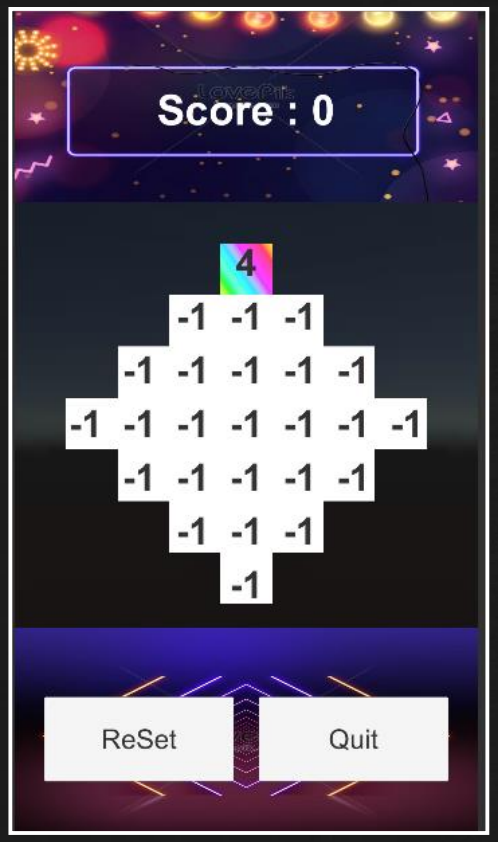
작성자 : 채용운
작성일 : 2019년 12월 27일

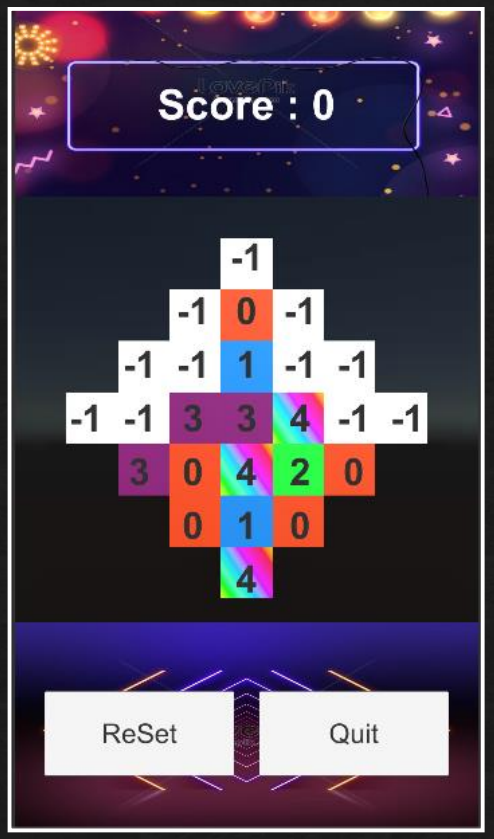
깃허브 주소 : https://github.com/chaeyongwoon/3MatchPuzzle

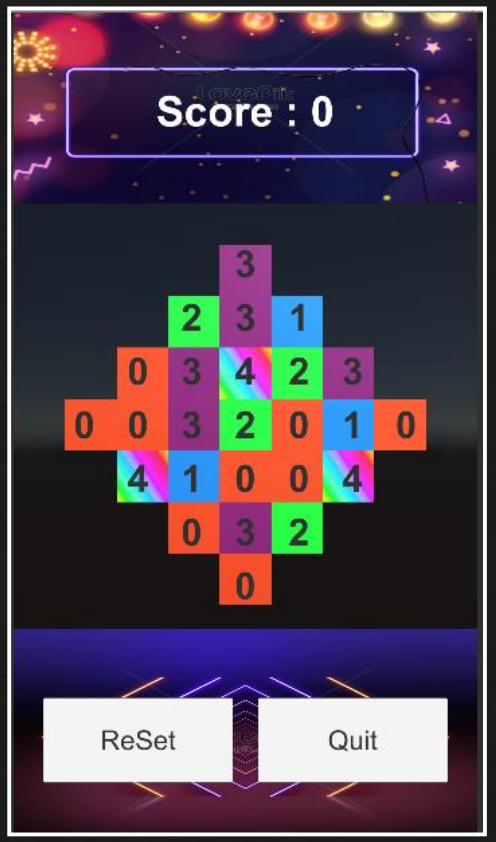
목차

1.프로젝트 설명
2.프로젝트 설계
3.프로젝트 구현

1. 프로젝트 설명



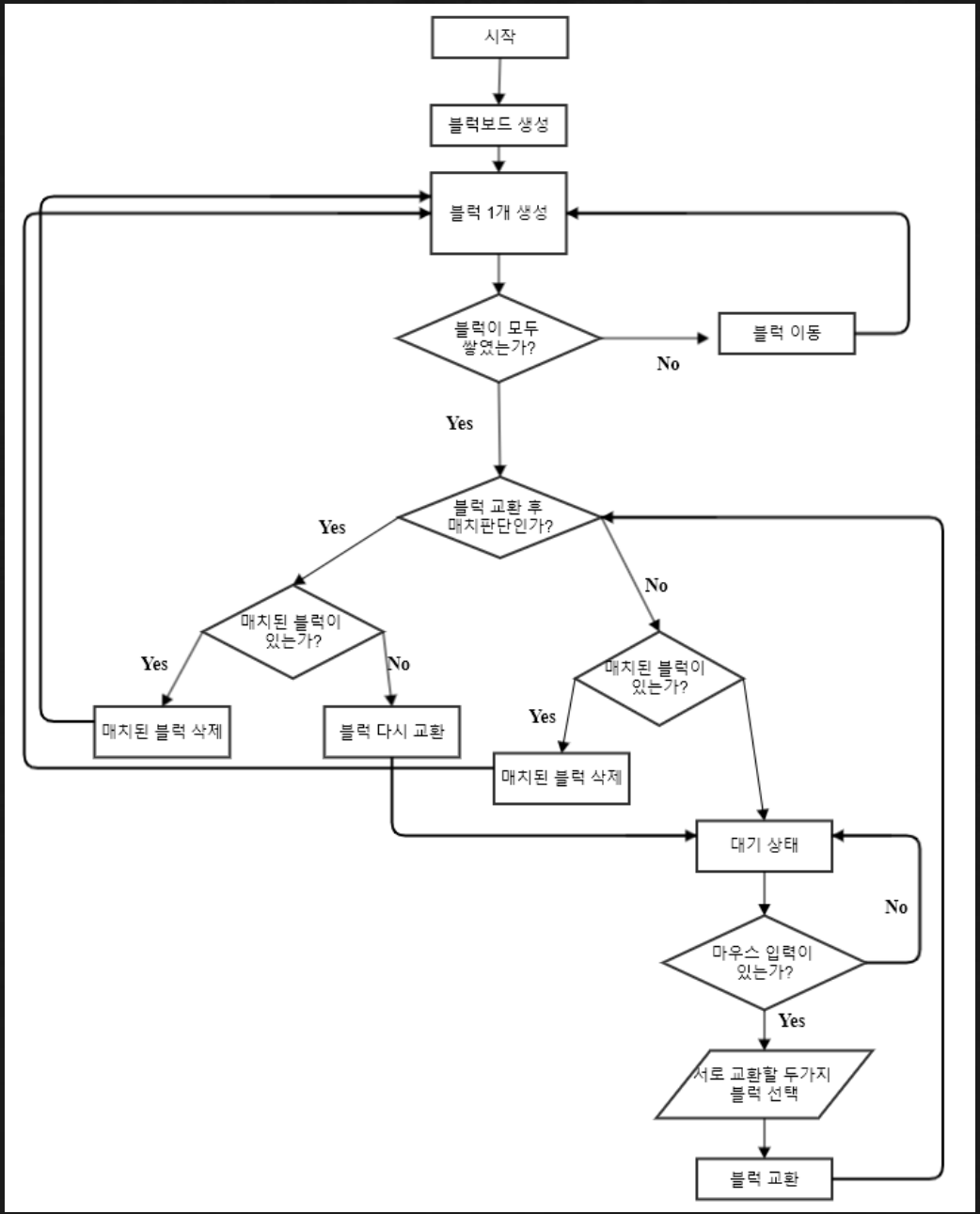




- 직선상에 같은 색상의 블록이 3개 이상 연결되면 매치되어 사라지는 게임.
- 블록은 한개씩 생성되며 좌,우 로 흘러내리며 쌓도록 구현.
- 무지개블록(4번)은 연속되어도 매치되지 않으며 매치되는 블록의 상,하,좌,우에 있을경우 회전상태 활성화.
- 회전상태인 무지개블록(4번)은 매치되는 블록의 상,하,좌,우에 있을경우 삭제.

2. 프로젝트 설계

게임 흐름도

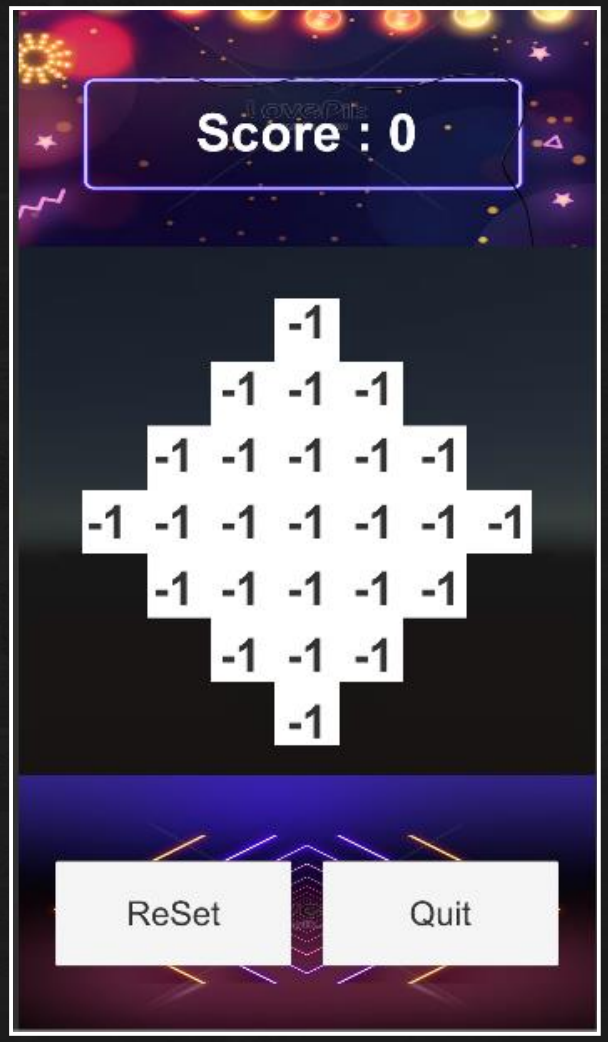


3. 프로젝트 구현

블록 보드 생성

- 시작시 블록 보드를 생성.
  (블록 보드의 값으로 매치판단)

- 빈공간은 -1로 표시되며 틀의 모양인
  검은 테두리 벽은 -2,-3 으로 설정.
  (서로 매치되지 않도록 바둑판 배열)




```
private void Awake()
{
    GmState = State.Create;
    Blockboard = new int[Width, Height];

    for (int x = 0; x < Width; x++)
    {
        for (int y = 0; y < Height; y++)
        {
            Blockboard[x, y] = -1;
        }
    }

    Blockboard[0, 0] = -2;
    Blockboard[0, 1] = -3;
    Blockboard[0, 2] = -2;
    Blockboard[0, 4] = -3;
    Blockboard[0, 5] = -2;
    Blockboard[0, 6] = -3;
    Blockboard[1, 0] = -3;
    Blockboard[1, 1] = -2;
    Blockboard[1, 5] = -3;
    Blockboard[1, 6] = -2;
    Blockboard[2, 0] = -2;
    Blockboard[2, 6] = -3;
    Blockboard[4, 0] = -3;
    Blockboard[4, 6] = -2;
    Blockboard[5, 0] = -2;
    Blockboard[5, 1] = -3;
    Blockboard[5, 5] = -2;
    Blockboard[5, 6] = -3;
    Blockboard[6, 0] = -3;
    Blockboard[6, 1] = -2;
    Blockboard[6, 2] = -3;
    Blockboard[6, 4] = -2;
    Blockboard[6, 5] = -3;
    Blockboard[6, 6] = -2;
}
```

3. 프로젝트 구현

함수의 실행

- 모든 함수는 Update( )에서 실행.
- State변수의 상태에 따라 함수가 실행.

```
private void Update()
{
    Createblock();          // 블럭 생성
    BlockMove();            // 블럭 하단이동 ( 하단, 좌측하단, 우측하단)
    BlockCheck();           // 블럭(보드) 매치체크
    BlockDelete();          // 팽이 큐브 활성화 판단 + 매치된 블럭 삭제

    MouseClick();           // 마우스 클릭 처리
    DragToMoveBlock();      // 마우스로 블럭 이동

    TextOn();               // UI 블럭(=보드)값 표시
}
```

```
void Createblock()
{
    if (GmState == State.Create)
    {
```

```
void BlockMove()
{
    if (GmState == State.Down)
    {
```

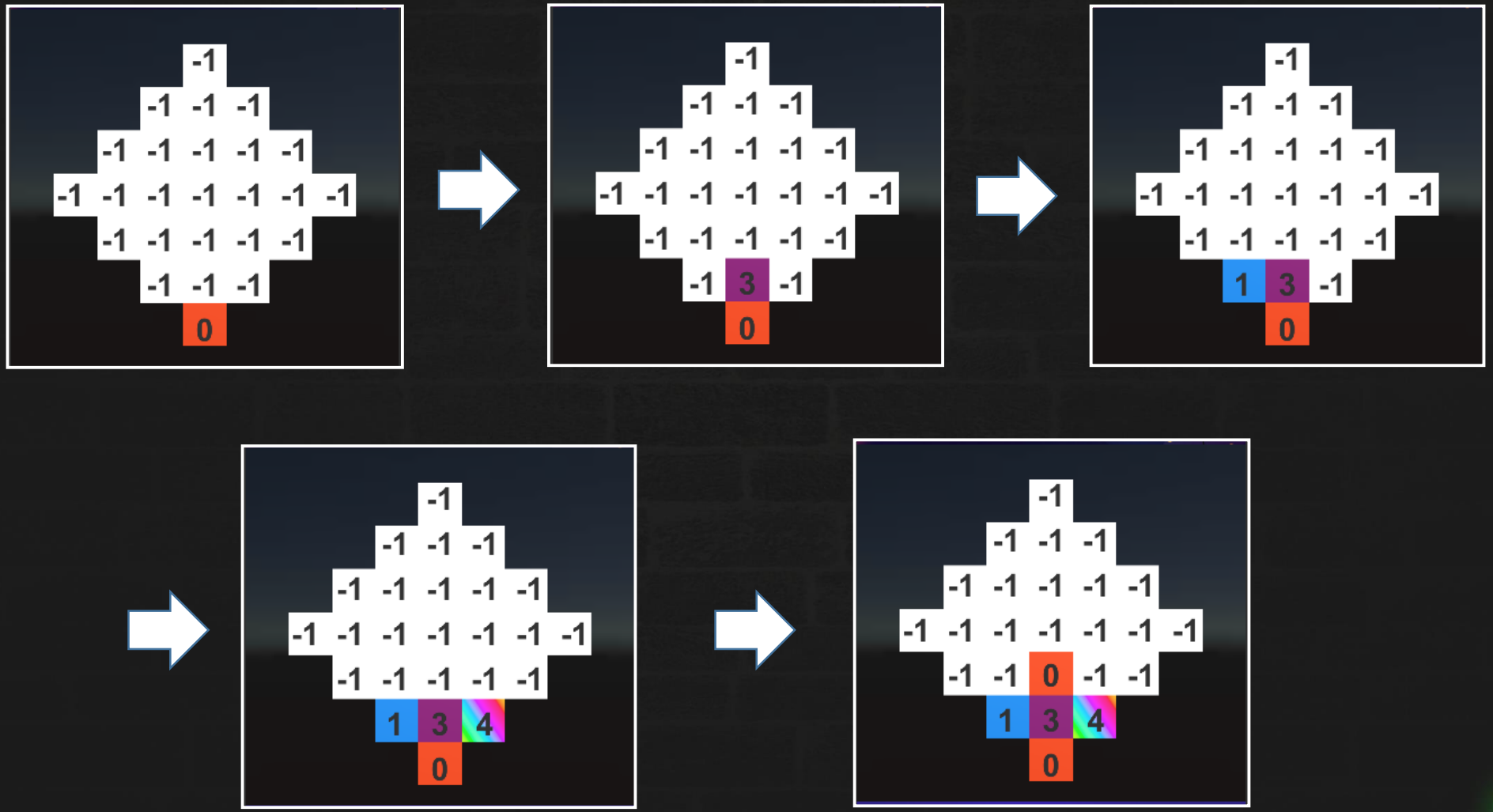
```
public enum State
{
    Create,
    Down,
    Move,
    Wait,
    Check,
}
public State GmState;
```

```
void BlockCheck()
{
    if ((GmState == State.Wait) || (GmState == State.Check))
    {
```

```
void BlockDelete()
{
    if ((GmState == State.Wait) || (GmState == State.Check))
    {
```

3. 프로젝트 구현

블록 생성 및 이동 - 1



- 블록 생성 후 이동, 좌우로 흘러내리면서 쌓이도록 구현

3. 프로젝트 구현

블록 생성 및 이동 - 2

```
void BlockMove()
{
    if (GmState == State.Down)
    {
        m_blocks = GameObject.FindGameObjectsWithTag("Block");
        if (m_blocks == null)
        {
            return;
        }

        foreach (GameObject block in m_blocks)
        {
            Block sblock = block.GetComponent<Block>();

            if (sblock.y == 0 || sblock.x == 0 || sblock.x == Width - 1)
            {
                GmState = State.Create;
                bReCheck = true;
            }
            else if (Blockboard[sblock.x, sblock.y - 1] == -1)
            {
                Blockboard[sblock.x, sblock.y - 1] = Blockboard[sblock.x, sblock.y];
                Blockboard[sblock.x, sblock.y] = -1;
                sblock.SetPos(sblock.x, sblock.y - 1);
                GmState = State.Down;
            }
            else if (Blockboard[sblock.x - 1, sblock.y - 1] == -1)
            {
                Blockboard[sblock.x - 1, sblock.y - 1] = Blockboard[sblock.x, sblock.y];
                Blockboard[sblock.x, sblock.y] = -1;
                sblock.SetPos(sblock.x - 1, sblock.y - 1);
                GmState = State.Down;
            }

            else if (Blockboard[sblock.x + 1, sblock.y - 1] == -1)
            {
                Blockboard[sblock.x + 1, sblock.y - 1] = Blockboard[sblock.x, sblock.y];
                Blockboard[sblock.x, sblock.y] = -1;
                sblock.SetPos(sblock.x + 1, sblock.y - 1);
                GmState = State.Down;

            }
            else if (sblock.y == Height - 1)
            {
                GmState = State.Wait;
            }
            else
            { bReCheck = true; }
        }
    }
}
```

- 하단,좌측하단,우측하단 순서로 빈 공간인지 판단 후 블록 이동
- 바닥부터 쌓여 꼭대기에 쌓일경우 GmState 상태를 Wait으로 전환

3. 프로젝트 구현

블록 매치판단

```
void BlockCheck()
{
    if ((GmState == State.Wait) || (GmState == State.Check))
    {

        for (int y = 0; y < Height; y++)
        {
            for (int x = 0; x < Width - 2; x++)
            {
                if (Blockboard[x, y] != 4)
                {
                    if (Blockboard[x, y] == Blockboard[x + 1, y])
                    {
                        if (Blockboard[x, y] == Blockboard[x + 2, y])
                        {
                            GameObject[] blocks = GameObject.FindGameObjectsWithTag("Block");

                            foreach (GameObject block in blocks)
                            {
                                Block sblock = block.GetComponent<Block>();

                                if (sblock.y != y)
                                    continue;

                                if (sblock.x == x && sblock.y == y)
                                {
                                    sblock.mState = Block.State.Match;
                                    continue;
                                }
                                if (sblock.x == x + 1 && sblock.y == y)
                                {
                                    sblock.mState = Block.State.Match;
                                    continue;
                                }
                                if (sblock.x == x + 2 && sblock.y == y)
                                {
                                    sblock.mState = Block.State.Match;
                                }
```

< 가로 매치 판단>

```
for (int x = 0; x < Width; x++)
{
    for (int y = 0; y < Height - 2; y++)
    {
        if (Blockboard[x, y] != 4)
        {
            if (Blockboard[x, y] == Blockboard[x, y + 1])
            {
                if (Blockboard[x, y] == Blockboard[x, y + 2])
                {
                    GameObject[] blocks = GameObject.FindGameObjectsWithTag("Block");

                    foreach (GameObject block in blocks)
                    {
                        Block sblock = block.GetComponent<Block>();

                        if (sblock.x != x)
                            continue;

                        if (sblock.x == x && sblock.y == y)
                        {
                            sblock.mState = Block.State.Match;
                            continue;
                        }
                        if (sblock.x == x && sblock.y == y + 1)
                        {
                            sblock.mState = Block.State.Match;
                            continue;
                        }
                        if (sblock.x == x && sblock.y == y + 2)
                        {
                            sblock.mState = Block.State.Match;
                        }
```

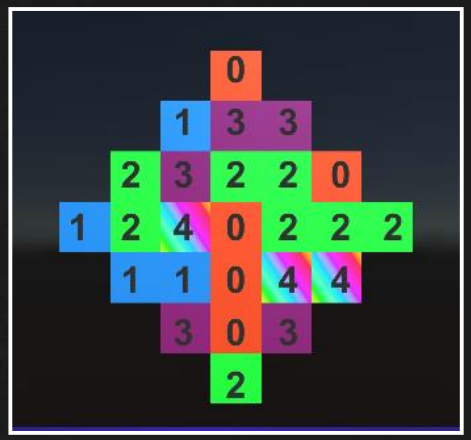
< 세로 매치 판단>

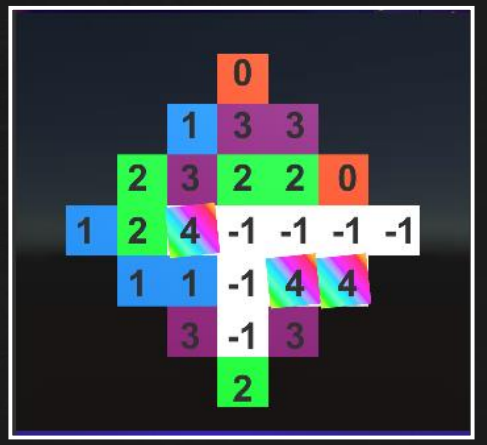
- 오른쪽(세로의 경우 위쪽)으로 이동하며 보드의 값을 비교하고 같을 경우에 블록의 상태를 Match 상태로 변경

3. 프로젝트 구현

매치된 블록 삭제

```
void BlockDelete()                          // 블럭 삭제 함수
{
    if ((GmState == State.Wait) || (GmState == State.Check))
    {

        GameObject[] blocks2 = GameObject.FindGameObjectsWithTag("Block");

        foreach (GameObject block2 in blocks2)
        {
            Block sblock2 = block2.GetComponent<Block>();

            if (sblock2.mState == Block.State.Match)
            {
                Blockboard[sblock2.x, sblock2.y] = -1;
                Destroy(block2);
                Score += 1;
                GmState = State.Down;
                bBlockReChange = false;


            }
        }
```





- 매치상태인 블록을 모두 찾아 블록을 삭제
- 삭제된 블록의 보드 값을 -1(빈공간) 으로 설정.

3. 프로젝트 구현

마우스 처리 -1

```
void MouseClick()
{
    if (GmState == State.Wait || GmState == State.Move)
    {
        if (Input.GetMouseButtonDown(0))
        {
            GmState = State.Move;
            RaycastHit Hit;
            MouseStartPos = Input.mousePosition;
            Ray ray = Camera.main.ScreenPointToRay(MouseStartPos);

            if (Physics.Raycast(ray, out Hit, Mathf.Infinity))
            {
                if (bDrag && Hit.collider.CompareTag("Block"))
                {
                    SelectBlock = Hit.collider.gameObject;
                    StartPos1 = SelectBlock.transform.position;
                }
            }
        }
```

```
        if (Input.GetMouseButton(0))
        {
            MouseEndPos = Input.mousePosition;
            MouseOffset = MouseStartPos - MouseEndPos;

            if (bDrag && SelectBlock != null)
            {
                if (MouseOffset.x > fMouseMoveDis)
                {
                    if (SelectBlock.transform.position.x > 0)
                        MouseDirection(-1, 0);
                }
                if (MouseOffset.x < -fMouseMoveDis)
                {
                    if (SelectBlock.transform.position.x < Width - 1)
                        MouseDirection(1, 0);
                }
                if (MouseOffset.y < -fMouseMoveDis)
                {
                    if (SelectBlock.transform.position.x < Height - 1)
                        MouseDirection(0, 1);
                }
                if (MouseOffset.y > fMouseMoveDis)
                {
                    if (SelectBlock.transform.position.y > 0)
                        MouseDirection(0, -1);
                }
```

- 좌클릭된 블록을 첫번째 블록(SelectBlock)으로 설정.
- 좌클릭된 상태에서 드래그한 방향에 따라 두번째 블록(TargetBlock)을 설정.

3. 프로젝트 구현

마우스 처리 -2

```
if (Input.GetMouseButtonUp(0))
{
    GmState = State.Check;

    if ((SelectBlock == null) || (TargetBlock == null))
    {
        GmState = State.Wait;
    }


    if (iswall)
    {
        bBlockReChange = false;

        SelectBlock = null;
        TargetBlock = null;

        bDrag = true;
        GmState = State.Wait;
        iswall = false;

    }

}
```

```
void MouseDirection(float _x, float _y)
{
    EndPos1 = new Vector3(StartPos1.x + _x, StartPos1.y + _y, 0);

    bDrag = false;

    GameObject[] blocks = GameObject.FindGameObjectsWithTag("Block");

    foreach (GameObject block in blocks)
    {
        Block sblock = block.GetComponent<Block>();

        if (sblock.x == EndPos1.x && sblock.y == EndPos1.y)
        {
            TargetBlock = block;
        }
        else if ((Blockboard[(int)EndPos1.x, (int)EndPos1.y] == -2)
            || (Blockboard[(int)EndPos1.x, (int)EndPos1.y] == -3))
        {
            iswall = true;
            return;
        }

    }
    StartPos2 = EndPos1;
    EndPos2 = StartPos1;
}
```

- 드래그 한 방향에 따라 두번째블록(TargetBlock) 설정.
- 드래그 한 방향이 벽(-2 , -3)일 경우 무효화 처리.
- 좌클릭을 떼었을때 상황에 따라 체크,대기 상태로 전환

3. 프로젝트 구현

블록 드래그 이동 - 1

```
void DragToMoveBlock()
{
    if (GmState == State.Move)
    {
        if (TargetBlock == null)
            return;


        if (TargetBlock.transform.position != StartPos1)
        {
            SelectBlock.transform.position = EndPos1;
            TargetBlock.transform.position = EndPos2;
            bBlockReChange = true;
            SwitchBoard();
        }


    }
}
```

```
void SwitchBoard()
{
    Block sBlock = SelectBlock.GetComponent<Block>();
    sBlock.x = (int)EndPos1.x;
    sBlock.y = (int)EndPos1.y;


    sBlock = TargetBlock.GetComponent<Block>();
    sBlock.x = (int)EndPos2.x;
    sBlock.y = (int)EndPos2.y;

    int Tmptype = Blockboard[(int)StartPos1.x, (int)StartPos1.y];
    Blockboard[(int)StartPos1.x, (int)StartPos1.y] = Blockboard[(int)EndPos1.x, (int)EndPos1.y];
    Blockboard[(int)EndPos1.x, (int)EndPos1.y] = Tmptype;
}
```

- 마우스 처리에서 선택된 두개 블록의 위치를 서로 교환.
- 그 후 보드값도 교환.
- bBlockReChange = true 설정. (후에 원래위치로 돌아올 경우 대비)

3. 프로젝트 구현

블록 드래그 이동 - 2

```
if (!MatchCheck())
{

    if (bBlockReChange)
    {
        Vector3 TmpStartPos = StartPos1;
        StartPos1 = StartPos2;
        StartPos2 = TmpStartPos;

        Vector3 TmpEndPos = EndPos1;
        EndPos1 = EndPos2;
        EndPos2 = TmpEndPos;

        ReChangeBlock();
        bBlockReChange = false;


        SelectBlock = null;
        TargetBlock = null;

        bDrag = true;
        GmState = State.Wait;
    }

    if (bReCheck)
    {
        bReCheck = false;
        bDrag = true;
    }

}
```

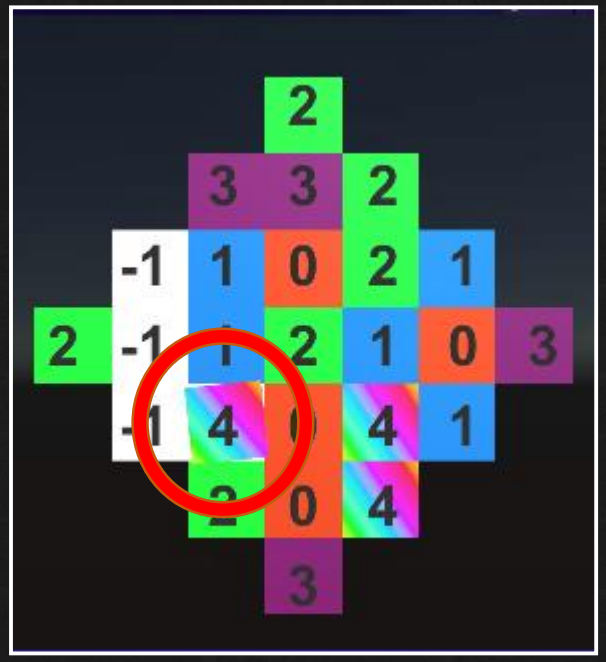
```
bool MatchCheck()
{
    GameObject[] blocks = GameObject.FindGameObjectsWithTag("Block");

    foreach (GameObject block in blocks)
    {
        if (block.GetComponent<Block>().mState == Block.State.Match)
            return true;
    }

    return false;
}
```

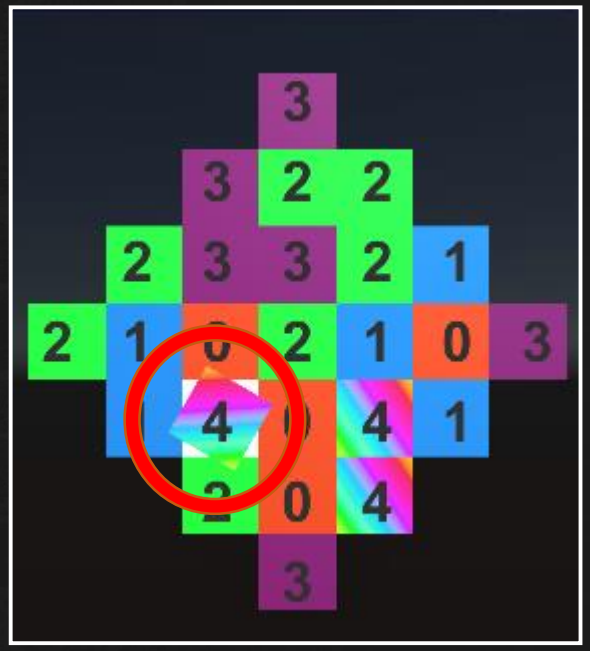
```
void ReChangeBlock()
{
    if (GmState == State.Check)
    {
        if (bBlockReChange && TargetBlock)
        {
            if (TargetBlock.transform.position != EndPos2)
            {
                SelectBlock.transform.position = EndPos1;
                TargetBlock.transform.position = EndPos2;

                SwitchBoard();
            }
            else
            {
                bBlockReChange = false;

                SelectBlock = null;
                TargetBlock = null;

                bDrag = true;
            }
```

- 블록 드래그 이동 후 매치판단.
- 드래그 이동 후 매치된게 없을경우 다시 원래위치로 이동 & 보드값 교환

3. 프로젝트 구현

특수블록 - 1





- 무지개블록(4번)은 직선상에 3개 이상이 모여도 매치되지 않음.
- 무지개블록이 매치된 블록의 상,하,좌,우 에 있을경우 회전 활성화.
- 회전이 활성화된 무지개블록이 다시 매치된 블록 주변에 있을경우 삭제.

3. 프로젝트 구현

특수블록 - 2

- 보드의 값이 4(무지개블록)일 경우 상,하,좌,우 에 있는 블록이 매치 상태이면 회전 활성화

- If(sblock1.x == x+1 && sblock.y == y)
- If(sblock1.x == x-1 && sblock.y == y)
- If(sblock1.x == x && sblock.y == y+1)
- If(sblock1.x == x && sblock.y == y-1)

(코드는 같은내용이 반복되어 중략했습니다)

```
for (int y = 0; y < Height; y++)
{
    for (int x = 0; x < Width; x++)
    {
        if (Blockboard[x, y] == 4)
        {
            string tmpname = string.Format("({0},{1})", x, y);
            tmpblock = GameObject.Find(tmpname).GetComponent<Block>();
            SpinCheck = false;

            GameObject[] blocks1 = GameObject.FindGameObjectsWithTag("Block");

            foreach (GameObject block1 in blocks1)
            {
                Block sblock1 = block1.GetComponent<Block>();

                if (sblock1.x == x + 1 && sblock1.y == y)
                {
                    if (sblock1.mState == Block.State.Match)
                    {
                        if ((SpinCheck == false) && (tmpblock.mState == Block.State.Stop))
                        {
                            tmpblock.mState = Block.State.Spin;
                            SpinCheck = true;
                        }
                        else if ((SpinCheck == false) && (tmpblock.mState == Block.State.Spin))
                        {
                            tmpblock.mState = Block.State.Match;
                            SpinCheck = true;
                        }
                    }
                }
```


Thank you