홍익대학교 게임소프트웨어

테트리스 기말 보고서

객체지향프로그래밍실습(2)

B693148 오혜미

2019-12-21

내용

추가한 클래스

클래스 이름	상속 클래스	간단 설명
Block	GameObject	Unity의 Prefab 느낌으로 구현하고 싶었다. Block Shapes라는 벡터를 구현하여 랜덤으로 모양을 갖도록 하였다. GameObect의 traverseStart 함수를 virtual로 선언하여 Block 클래스에서의 decideRandomShape 함수를 traverseStart에서 실행하도록 하였다.
BlokcScript	Component	GridManager에서 정보를 얻어와 Block의 사용자입력과 이동, 회전 및 배치를 담당하였다. GameObject에 Transform 컴포넌트가 자동으로 들어가 있듯, Block에 BlockScript가 자동으로 들어가도록 설계하였다.
BlockSpawnerScript	Component	newBlock을 선언하여 start함수에서 생성하여주었다. 새로운 블록들을 만들지 않고 block의 shape정보만 교체하여 재활용하였다. Next block 정보를 보여주는 showNextBlock 함수를 만들어 정보를 볼 수 있도록 하였다.
ScoreScript	Component	점수를 관리하는 클래스이다.
GameUIScript	Component	점수, 게임오버, 라인 지우기의 화면을 출력해주는 클래스이다.

추가한 함수

클래스 이름	함수	간단 설명

Block	void decideRandomShape() void traverseStart()	-랜덤하게 블록모양을 정해줌 -decideRandomShape함수가 실행되는 곳
BlockScript	void move() void rotateShape() void rotateBlock() bool place() void resetSpeed()	-키보드를 입력 받아 블록을 좌우로 움직임 -모양을 회전시킴 -블록을 회전시킴(회전 가능한 지 확인한 후) -블록이 놓아졌는지 확인하고 해당 블록을 비활성화 시킴 -스피드를 원래 속도로 되돌림
BlockSpawner	void setNeedNewBlock(bool need) void showNextBlock();	-새로운 블록이 필요한지 아닌지를 알려주는 변수를 set하는 함수 -다음 블록을 보여줌

GridManager	bool isInsideRight(Transform* transform)
	bool isInsideLeft(Transform* transform)
	bool isGrounded(Transform* transform)
	void place(Transform * transform)
	voiddropBlock(Transform * transform);
	Vector2 getPosition()
	bool getGameOver()
	
	void checkFullyOcuppiedLine()
	void eraseFullyOcuppiedLine(int line);
	void dropLines(int line)
	
	boolcheckCollisionRight(Transform* transform)
		-맵에서 오른쪽으로 나갔는지 확인
		-맵에서 왼쪽으로 나갔는지 확인
		
		-block이 놓였는지 확인
		
		-놓인 block을 map에 정보 입력
		-block을 단숨에 떨어뜨려줌
		
		-포지션을 transform에서 get
		-게임 오버의 여부를 get
		
		-꽉차있는 라인이 있나 확인하고 eraseFullyOcuppiedLine 함수를 호출하여 지운 후 dropLines 함수를 호출하여 남은 블록들을 내림.
		
		-다른 블록이 오른쪽에 있나 확인
		-다른 블록이 왼쪽에 있나 확인

	bool checkCollisionLeft(Transform* transform) void drawMap();	-맵을 그림(맵의 내용 및 테두리)
Transform	Vector2 getScale()const void setScale(const Vector2& scale) string getShape() void setShape(const string shape)	-크기(width, height)를 get -크기를 set -shape를 get -shape를 set
Utils	void drawBlock(const char* shape, int w, int h, const Vector2& pos)	-블록을 그릴 때 비어있는 공간은 그리지 않음(블록 때문에 배경이나 다른 블록이 보이지 않는 것을 방지)
ScoreScript	void addScore() void addScore(int score) int getScore() Vector2 getPosition()	-점수 +1 -점수+score -점수를 get -포지션을 get
GameUIScript	void showGameOver() void showLineClear() void showScore()	-게임 오버가 되면 게임오버 글을 화면에 띄움 -라인을 지울 때 Line Clear를 화면에 띄움 -실시간으로 점수를 보여줌

구현 내용

블록 충돌 체크

```
105  //오른쪽 충돌 확인
106 bool GridManager::checkCollisionRight(Transform * transform)
107  {
108      for (int h = transform->getScale().y - 1; h >= 0; h--)
109      {
110          for (int w = 0; w < transform->getScale().x; w++)
111          {
112              //블록(' '이 아닌 '\xB0')이고 그 다음 칸이 true로 차있을 때
113              if (transform->getShape().c_str()[h * (int)transform->getScale().x + w] == '\xB0'
114                  && map[((int)transform->getPosition().y + h) * width + (int)transform->getPosition().x + w + 1])//블록 다음 자리(오른쪽)
115                  return true;
116          }
117      }
118      return false;
119  }
120
121  //왼쪽 충돌확인
122 bool GridManager::checkCollisionLeft(Transform * transform)
123  {
124      for (int h = transform->getScale().y - 1; h >= 0; h--)
125      {
126          for (int w = 0; w < transform->getScale().x; w++)
127          {
128              //블록(' '이 아닌 '\xB0')이고 그 이전 칸이 true로 차있을때
129              if (transform->getShape().c_str()[h * (int)transform->getScale().x] == '\xB0'
130                  && map[((int)transform->getPosition().y + h) * width + (int)transform->getPosition().x + w - 1])//블록 이전 자리(왼쪽)
131                  return true;
132          }
133      }
134      return false;
135  }
```

블록의 각 칸마다 확인.

블록 맵 탈출 방지

```
136
137  //오른쪽 확인 맵의 안에 있는지
138 bool GridManager::isInsideRight(Transform* transform)
139  {
140      if (transform == nullptr)
141          return false;
142      for (int h = 0; h < transform->getScale().y; h++)
143      {
144          for (int w = transform->getScale().x - 1; w >= 0; w--)
145          {
146              if (transform->getShape().c_str()[h * (int)transform->getScale().x + w ] == '\xB0'
147                  && transform->getPosition().x + w+1  >= position.x + width-1)
148                  return false;
149          }
150      }
151      return true;
152  }
153  //왼쪽 확인 맵의 안에 있는지
154 bool GridManager::isInsideLeft(Transform* transform)
155  {
156      if (transform == nullptr)
157          return false;
158
159      int x = transform->getPosition().x;
160      if (x  < position.x)
161          return false;
162
163      return true;
164  }
165
```

블록 회전

```
63  //모양을돌리기
64  void BlockScript::rotateShape()
65  {
66      int w = transform->getScale().x;
67      int h = transform->getScale().y;
68      static char* shape = new char[w* h+1];
69      for (int y = 0; y < h; y++)
70          for (int x = 0; x < w; x++)
71              shape[(w - 1 - x)*h + y ] = transform->getShape()[y*w+ x];
72      transform->setShape(shape);
73      transform->setScale(Vector2(h, w));
74  }
75  //블록 돌리기
76  void BlockScript::rotateBlock()
77  {
78      //돌릴 상황이 되지않을 경우 원래로 돌아가기 위해 저장
79      Block* saveBlock = new Block();
80      saveBlock->getTransform()->setPosition(transform->getPosition());
81      saveBlock->getTransform()->setScale(transform->getScale());
82      saveBlock->getTransform()->setShape(transform->getShape());
83      //모양을 돌림
84      rotateShape();
85      //왼쪽 맵 밖으로 나가는 지 확인하하고 나가면 오른쪽 칸을 확인 후 한칸 오른쪽으로 이동함
86      if (!gm->isInsideLeft(transform))
87      {
88          if (!gm->checkCollisionRight(transform))
89              transform->setPosition(transform->getPosition() + Vector2(1, 0));
90      }
91      //오른쪽 맵 밖으로 나가는 지 확인하하고 나가면 왼쪽 칸을 확인 후 한칸 왼쪽으로 이동함
92      while (!gm->isInsideRight(transform))
93      {
94          if (!gm->checkCollisionLeft(transform))
95              transform->setPosition(transform->getPosition() + Vector2(-1, 0));
96      }
97      //돌린것을 되돌림(로테이션 실패)
98      if (!gm->isInsideRight(transform) || gm->checkCollisionRight(transform))
99      {
100         transform->setPosition(saveBlock->getTransform()->getPosition());
101         transform->setScale(saveBlock->getTransform()->getScale());
102         transform->setShape(saveBlock->getTransform()->getShape());
103     }
104
105     delete saveBlock;
106 }
```

실제 테트리스 게임처럼 블록을 돌릴 수 있도록 함. 공간이 있을 경우에는 회전. 회전 후 맵을 빠져나가거나 다른 블록이랑 겹칠 경우 블록이 회전할 수 있을 때까지 이동해줌.(블록의 모양이 완전히 다른 것들과 겹치지 않도록, 왼쪽의 경우는 회전할 때 절대 다른 블록들과 겹치지 않기 때문에 고려하지 않고 맵밖을 빠빠져나갈 때 고려하여 이동시켜 줄 때만 이동 방향의 충돌을 확인. 반면에 오른쪽은 겹치는 경우가 많기 때문에 다른 블록과의 충돌을 계속 확인해 주어 충돌하지않게 이동시켜줘야함.) 확인 후 돌려서는 안되는 상황이라면(맵을 빠져나갈 경우, 다른 블록과 겹쳐질 경우) 다시 원래의 모양으로 되돌림.

스코어

```
void ScoreScript::start()
{
    GridManager* map = static_cast<GridManager *>(gameObject->getComponent<GridManager>());
    position = map->getPosition() + map->getWidth() + Vector2(2, 10);
}

void ScoreScript::update()
{
}

void ScoreScript::addScore()
{
    score++;
}

void ScoreScript::addScore(int score)
{
    this->score += score;
}
```

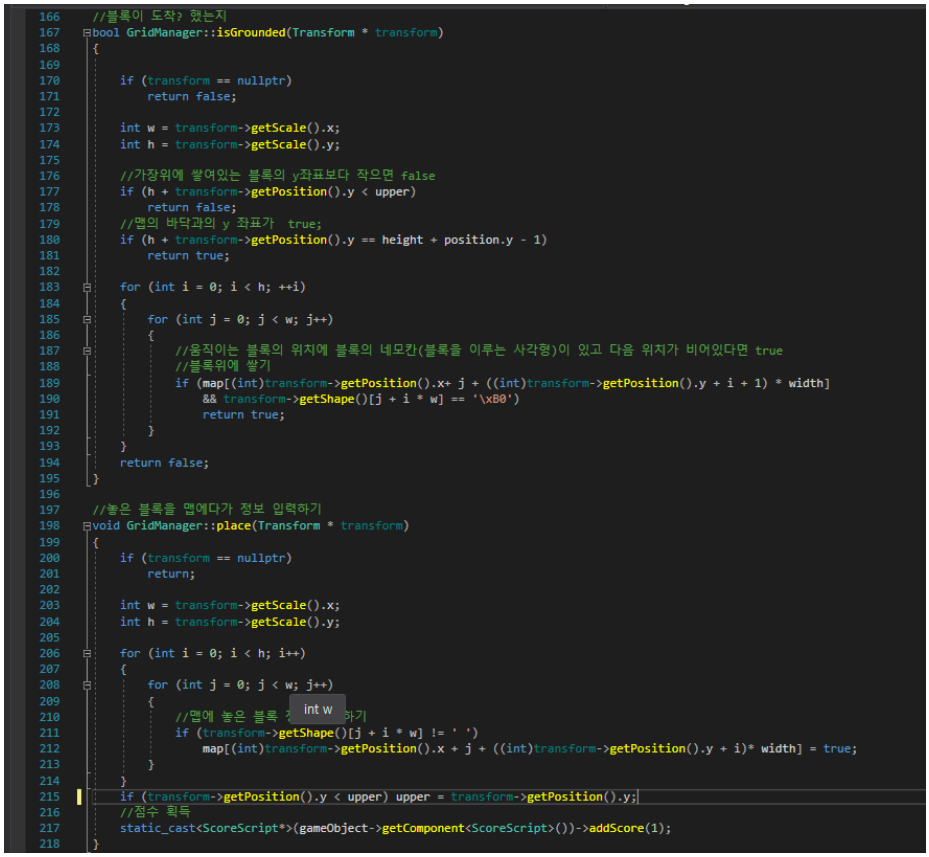
블록 지우기

```
//꽉 찬 라인을 찾고 지우고 나머지 블록들을 내려줌
void GridManager::checkFullyOcuppiedLine()
{

    for (int h = 0; h < height; h++)
    {
        for (int w = 0; w < width; w++)
        {
            //false가 있으면 브레이크
            if (!map[h * width + w])
                break;
            if (w == width - 1)
            {
                eraseFullyOcuppiedLine(h);
                dropLines(h);
            }
        }

    }
}
// 라인 지우기 함수
void GridManager::eraseFullyOcuppiedLine(int line)
{

    //맵을 false로 설정하기
    for (int w = 0; w < width; w++)
        map[line * width + w] = false;
    static_cast<ScoreScript*>(gameObject->getComponent<ScoreScript>())->addScore(10);
    clearLine = true;

}
// 남은 블록들 내리기
void GridManager::dropLines(int line)
{
    for (int i = line-1; i >= upper-1; i--)
    {
        for (int j = 0; j < width; j++)
        {
            //윗줄을 아랫줄로
            map[(i + 1) * width + j] = map[i * width + j];
        }
    }
    upper++;//upper는 한 줄 증가 (줄이 하나 사라져 y값이 증가했기때문)
}
```

블록의 라인이 꽉 차면(map의 값으로 확인) 블록을 지우고(map의 해당 값을 false로 변경) 그 윗 줄을 한

칸씩 내림(map에서 바로 윗줄을 자신에게 복사). 지울 때는 10점의 점수를 더해줌.

블록 놓기

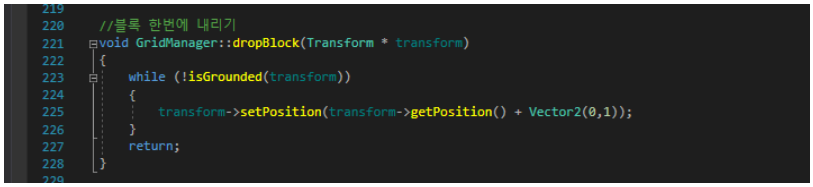


```
166   //블록이 도착? 했는지
167   bool GridManager::isGrounded(Transform * transform)
168   {
169
170       if (transform == nullptr)
171           return false;
172
173       int w = transform->getScale().x;
174       int h = transform->getScale().y;
175
176       //가장위에 쌓여있는 블록의 y좌표보다 작으면 false
177       if (h + transform->getPosition().y < upper)
178           return false;
179       //맵의 바닥과의 y 좌표가  true;
180       if (h + transform->getPosition().y == height + position.y - 1)
181           return true;
182
183       for (int i = 0; i < h; ++i)
184       {
185           for (int j = 0; j < w; j++)
186           {
187               //움직이는 블록의 위치에 블록의 네모칸(블록을 이루는 사각형)이 있고 다음 위치가 비어있다면 true
188               //블록위에 쌓기
189               if (map[(int)transform->getPosition().x+ j + ((int)transform->getPosition().y + i + 1) * width]
190                   && transform->getShape()[j + i * w] == '\xB0')
191                   return true;
192           }
193       }
194       return false;
195   }
196
197   //놓은 블록을 맵에다가 정보 입력하기
198   void GridManager::place(Transform * transform)
199   {
200       if (transform == nullptr)
201           return;
202
203       int w = transform->getScale().x;
204       int h = transform->getScale().y;
205
206       for (int i = 0; i < h; i++)
207       {
208           for (int j = 0; j < w; j++)
209           {
210               //맵에 놓은 블록 정보 입력하기
211               if (transform->getShape()[j + i * w] != ' ')
212                   map[(int)transform->getPosition().x + j + ((int)transform->getPosition().y + i)* width] = true;
213           }
214       }
215       if (transform->getPosition().y < upper) upper = transform->getPosition().y;
216       //점수 획득
217       static_cast<ScoreScript*>(gameObject->getComponent<ScoreScript>())->addScore(1);
218   }
```

블록이 이동할 다음 칸을 비교 및 좌표를 비교하여 블록을 놓음.

블록이 놓아질 때마다 1점을 얻는다.

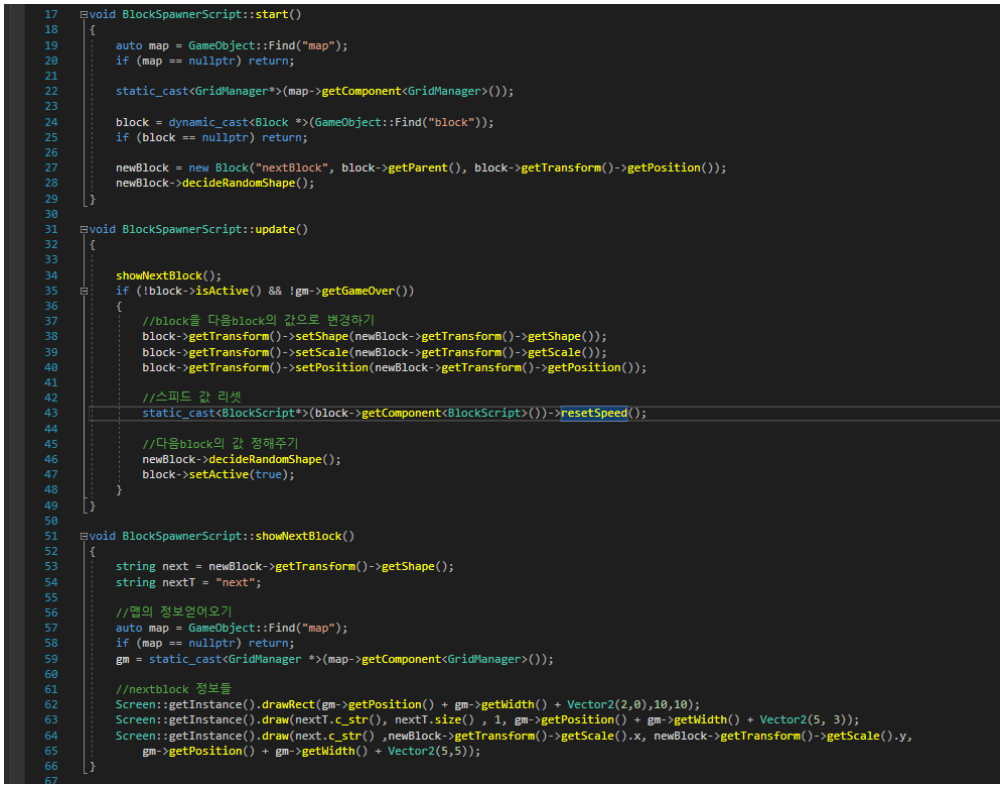
블록 단숨에 놓기



```
219
220   //블록 한번에 내리기
221   void GridManager::dropBlock(Transform * transform)
222   {
223       while (!isGrounded(transform))
224       {
225           transform->setPosition(transform->getPosition() + Vector2(0,1));
226       }
227       return;
228   }
229
```

블록이 놓아지는 포지션까지 계속하여 1을 더해 준 후 위치 업데이트

블록 스폰



```
void BlockSpawnerScript::start()
{
    auto map = GameObject::Find("map");
    if (map == nullptr) return;

    static_cast<GridManager*>(map->getComponent<GridManager>());

    block = dynamic_cast<Block *>(GameObject::Find("block"));
    if (block == nullptr) return;

    newBlock = new Block("nextBlock", block->getParent(), block->getTransform()->getPosition());
    newBlock->decideRandomShape();
}

void BlockSpawnerScript::update()
{

    showNextBlock();
    if (!block->isActive() && !gm->getGameOver())
    {
        //block을 다음block의 값으로 변경하기
        block->getTransform()->setShape(newBlock->getTransform()->getShape());
        block->getTransform()->setScale(newBlock->getTransform()->getScale());
        block->getTransform()->setPosition(newBlock->getTransform()->getPosition());

        //스피드 값 리셋
        static_cast<BlockScript*>(block->getComponent<BlockScript>())->resetSpeed();

        //다음block의 값 정해주기
        newBlock->decideRandomShape();
        block->setActive(true);
    }
}

void BlockSpawnerScript::showNextBlock()
{
    string next = newBlock->getTransform()->getShape();
    string nextT = "next";

    //맵의 정보얻어오기
    auto map = GameObject::Find("map");
    if (map == nullptr) return;
    gm = static_cast<GridManager *>(map->getComponent<GridManager>());

    //nextblock 정보들
    Screen::getInstance().drawRect(gm->getPosition() + gm->getWidth() + Vector2(2,0),10,10);
    Screen::getInstance().draw(nextT.c_str(), nextT.size() , 1, gm->getPosition() + gm->getWidth() + Vector2(5, 3));
    Screen::getInstance().draw(next.c_str() ,newBlock->getTransform()->getScale().x, newBlock->getTransform()->getScale().y,
        gm->getPosition() + gm->getWidth() + Vector2(5,5));
}
```

objs에는 하나의 block만을 넣어 두고 블록 스포너에서 next block을 선언하고 생성하여 block 이 place되어 enabled = false 가 되었을 때 block에게 next block의 shape와 position(스폰 지점), scale(shape정보에 필요한 w, h값)을 넣어 줌. 다음에 나올 block 은 또다시 새로운 모양을 넣어줌.(블록 재활용) 이와 더불어 다음 블록의 모양을 사용자에게 보여줌.

블록 컨트롤

```
24
25  void BlockScript::update()
26  {
27      if (place())
28          return;
29
30      if (speed <= downTime)
31      {   //맵 안에 있고 다른 블록들과 부딪히지않을 때 이동가능
32          if (Input::GetKeyDown(KeyCode::Right))
33          {
34
35              if (gm->isInsideRight(transform) && !gm->checkCollisionRight(transform))
36                  transform->setPosition(transform->getPosition() + Vector2::right);
37          }
38
39          if (Input::GetKeyDown(KeyCode::Left))
40          {
41              if (gm->isInsideLeft(transform) && !gm->checkCollisionLeft(transform))
42                  transform->setPosition(transform->getPosition() + Vector2::left);
43          }
44
45          if (Input::GetKeyDown(KeyCode::Up)) // 키보드 윗 키: 로테이션
46              rotateBlock();
47
48          if (Input::GetKeyDown(KeyCode::Down))// 키보드 아래키 :속도 증가 시키기
49              speed += 30;
50
51          if (Input::GetKeyDown(KeyCode::Space)) //키보드 스페이스 :단숨에 블록 놓기
52              gm->dropBlock(transform);
53          downTime--;
54      }
55      if (speed > downTime)
56      {
57          //0.1초마다 블록들을 정확하게 한 칸씩 내림.
58          transform->setPosition(Vector2(transform->getPosition().x, transform->getPosition().y + 1.0));
59          downTime = 100;
60      }
61  }
```

블록 충돌, 맵을 체크 후 통해 양 옆 이동 가능.

블록 충돌, 맵을 체크 후 회전 가능.

스피드를 설정하여 키보드 아래 키를 누르면 스피드를 빠르게 조절 가능.

다운 타임을 설정하여 블록이 정확히 1 씩 이동 할 수 있도록 하여 소수점으로 이동하면 생길 수 있는 문제를 방지함.

어려웠던 점

1. 블록 스폰: 과제를 진행하며 가장 오류가 많이 났던 부분으로 블록이 겹쳐져 온 경우, 블록이 스폰지역에서 내려오지 않는 경우 등의 오류를 겪었음.
2. Line clear를 화면에 띄울 때 시간주기: 시간 내에 사라지지 않고 계속 보이거나 아예 보이지 않는 오류를 겪었음.
3. 블록 회전: 하면서 가장 힘들었던 부분으로 블록을 이동시키는 아이디어를 생각해 내는데 오래 걸렸음. 1자 모양 블록이 맵 밖으로 나오는 오류도 있었고 회전하면 블록이 겹치는 오류도 겪었음.

최종 소감

전에 했던 과제에서 나타났던 많은 오류들을 이번 과제를 통해서 다시 한 번 생각하며 오류를 해결하며 조금 더 생각을 키울 수 있던 과제였음. 또한 컴포넌트 기반 설계에 대한 이해를 키울 수 있었음.