인공지능과 게임프로그래밍

유형	AI game
교수님	이대호
학년/학기	3학년 1학기

코드에 대한 설명은 코드에 안의 주석으로 설명해놨습니다.

약간의 쉬운 내용이나 중요도가 낮은 내용들은 코드가 아예 없거나 주석이 없습니다.

기본 레이어 (공, 패달, 벽돌) 구현

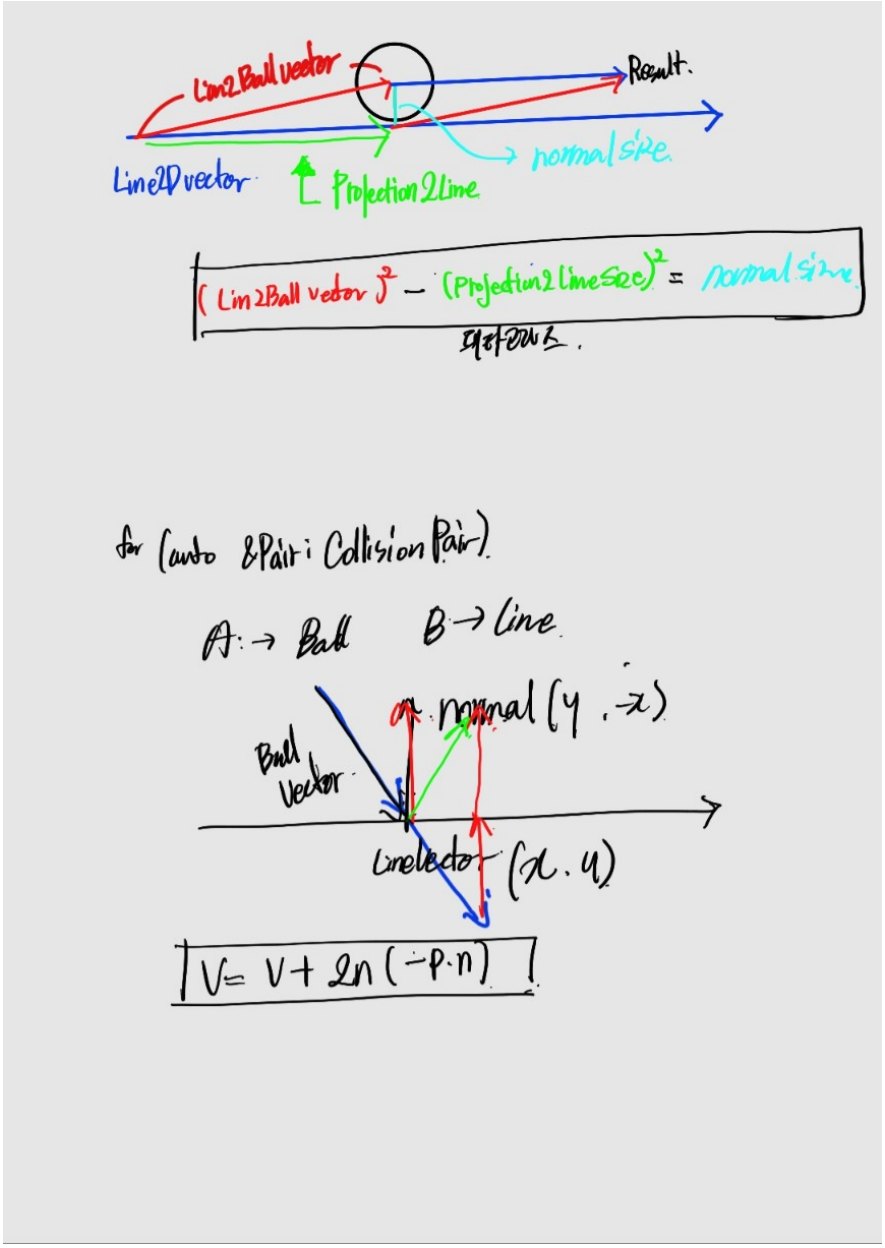
```
Circle1 = new CKhuGleSprite(GP_STYPE_ELLIPSE, GP_CTYPE_DYNAMIC, CKgLine(CKgPoint(360, 360), CKgPoint(380, 380)), KG_COLOR_24_RGB(255,

  Paddle1 = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_DYNAMIC, CKgLine(CKgPoint(300, 400), CKgPoint(400, 400)), KG_COLOR_24_RGB(0, 255
  Paddle2 = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_DYNAMIC, CKgLine(CKgPoint(350, 385), CKgPoint(350, 415)), KG_COLOR_24_RGB(0, 255

  OutLine1 = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_STATIC, CKgLine(CKgPoint(-1, 0),   CKgPoint(600, 0)  ), KG_COLOR_24_RGB(255, 25
  OutLine2 = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_STATIC, CKgLine(CKgPoint(600, 0), CKgPoint(600, 425)), KG_COLOR_24_RGB(255, 255
  OutLine3 = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_STATIC, CKgLine(CKgPoint(-1, 425), CKgPoint(600, 425)), KG_COLOR_24_RGB(255, 0,
  OutLine4 = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_STATIC, CKgLine(CKgPoint(-1, 0),   CKgPoint(-1, 425)  ), KG_COLOR_24_RGB(255, 2

  m_pGameLayer->AddChild(Circle1);
  m_pGameLayer->AddChild(Paddle1);
  m_pGameLayer->AddChild(Paddle2);
  m_pGameLayer->AddChild(OutLine1);
  m_pGameLayer->AddChild(OutLine2);
  m_pGameLayer->AddChild(OutLine3);
  m_pGameLayer->AddChild(OutLine4);
  Circle1->m_Velocity = CKgVector2D(-200, 200);

  // 벽돌 생성
  brick_num = 0;
  for (int column = 0; column < 4; column++) {
    for (int row = 0; row < 6; row++) {
      Brick1[brick_num] = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_KINEMATIC, CKgLine(CKgPoint(100*row,    40*column+20), CKgPoint(10
      Brick2[brick_num] = new CKhuGleSprite(GP_STYPE_LINE, GP_CTYPE_KINEMATIC, CKgLine(CKgPoint(100*row+50,  40*column   ), CKgPoint(1
      m_BrickContainer.push_back(Brick1[brick_num]);
      m_BrickContainer.push_back(Brick2[brick_num]);
      m_pGameLayer->AddChild(Brick1[brick_num]);
      m_pGameLayer->AddChild(Brick2[brick_num]);
      brick_num++;
    }
  }
```

키보드 컨드롤러

```
/* 키보드 컨트롤러*/
  if (m_bKeyPressed[VK_LEFT])
  {
    if (Paddle1->m_Center.x >= 55 && Paddle1->m_Center.x <= 545)
    {
      Paddle1->MoveBy(-2, 0);
      Paddle2->MoveBy(-2, 0);
    }
    else if (Paddle1->m_Center.x >= 545)
    {
      Paddle1->MoveBy(-2, 0);
      Paddle2->MoveBy(-2, 0);
    }
  }
  if (m_bKeyPressed[VK_RIGHT])
  {
    if (Paddle1->m_Center.x > 55 && Paddle1->m_Center.x < 545)
    {
      Paddle1->MoveBy(2, 0);
      Paddle2->MoveBy(2, 0);
```

```
    }
    else if (Paddle1->m_Center.x <= 55)
    {
      Paddle1->MoveBy(+2, 0);
      Paddle2->MoveBy(+2, 0);
    }
  }
  /* ESC를 눌러 종료 */
  if (m_bKeyPressed[VK_ESCAPE])
  {
    EndGame();
  }
```

충돌된 물체에 대해서 충돌 구현 1. 충돌 확인 2. 충돌 구현



충돌 확인 및 구현

```
else if (static_cast<CKhuGleSprite*>(Target)->m_nType == GP_STYPE_LINE)
{
  /*선백터를 구하고 이를 정규화함 -> 나중에 내적에서 사용하기 위해서임*/
  /*추가로 해당 선의 벡터의 크기를 구함*/
  CKgVector2D LineVecotr = {static_cast<double>(Target->m_lnLine.End.X - Target->m_lnLine.Start.X), static_cast<double>(Target->m_lnLi
  LineVecotr.Normalize();
  double LineVecotrSize = sqrt(pow(static_cast<double>(Target->m_lnLine.End.X - Target->m_lnLine.Start.X), 2) + pow(static_cast<double

  /*볼과 선의 백터와 크기를 구함*/
  CKgVector2D Line2BallVector = Ball->m_Center - Target->m_lnLine.Start;

  /*내적한 백터를 구하고 크기를 구함*/
  CKgVector2D Projection2Line = Line2BallVector.Dot(LineVecotr) * LineVecotr;
  double Projection2LineSize = Line2BallVector.Dot(LineVecotr);

  /*법선 백터의 크기를 구함*/
  double NormalSize = sqrt(pow(CKgVector2D::abs(Line2BallVector), 2) - pow(Projection2LineSize, 2));
  if (Ball->m_Radius >= NormalSize - Target->m_nWidth/2)
  {
    /*두 백터의 합은 결국 최종 이동값의 크기와 같음*/
    CKgVector2D ReflectionVector = Projection2Line + Line2BallVector;
    /*충돌 여부 확인 벡터(교수님 설명 참고)*/
    if (Line2BallVector.Dot(LineVecotr) >= 0 && Line2BallVector.Dot(LineVecotr) <= LineVecotrSize)
    {
      CollisionPairs.push_back({Ball, Target});
      /*충돌된 크기를 기준으로  충돌하는 크기로 나누어주어서 충돌량을 계산함*/
      double impact = NormalSize - Target->m_nWidth/2;
      Ball->MoveBy(ReflectionVector.x * impact / CKgVector2D::abs(ReflectionVector), ReflectionVector.y * impact / CKgVector2D::abs(Re
    }
}
```

충돌된 스프라이트 가져와서 충돌 구현

```
for(auto &Pair : CollisionPairs)
    {
      /*선과의 충돌*/
      if (Pair.second->m_nType == GP_STYPE_LINE)
      {
        CKhuGleSprite* Ball = Pair.first;
        CKhuGleSprite* Line = Pair.second;

        /*법선 백터를 구하기 위해 구하는 충돌한 선 백터*/
        CKgVector2D LineVector = {static_cast<double>(Line->m_lnLine.Start.X - Line->m_lnLine.End.X), static_cast<double>(Line->m_lnLi
        LineVector.Normalize();
        /*법선 벡터*/
        CKgVector2D normal = { LineVector.y, -LineVector.x };
        /*법선 백터를 기준으로 반사각 구하는 공식 진행*/
        Ball->m_Velocity = Ball->m_Velocity - 2 * (Ball->m_Velocity.Dot(normal)) * normal;

        /*그냥 매번 똑같이 튀기면 재미 없으니까 약간의 랜덤한 값 추가해서 게임이 매판 다른 게임이 될 수 있게 함.*/
        double randX = ((double)rand() / RAND_MAX) * 0.2 - 0.1; // Random value between -0.1 and 0.1
        double randY = ((double)rand() / RAND_MAX) * 0.2 - 0.1; // Random value between -0.1 and 0.1

        CKgVector2D randomVector = { randX, randY };
        Ball->m_Velocity += randomVector;
        if (Line == OutLine3)
        {
          life -= 1;
          if (life <= 0)
          {
            life = 0;
            Ball->m_Velocity.x = 0;
            Ball->m_Velocity.y = 0;
          }
        }
        if (Pair.second->m_nCollisionType == GP_CTYPE_KINEMATIC)
        {
          for (auto& line : m_BrickContainer)
          {
            if (Line->m_Center.x == line->m_Center.x && Line->m_Center.y == line->m_Center.y)
              line->m_nType = GP_STYPE_DESTORY;
          }
          Line->m_nType = GP_STYPE_DESTORY;
          score += 10;
        }
      }
    }
  }
```

화면에 제목, 생명, 점수 표시

```
m_pScene->Render();
  DrawSceneTextPos("Lab 01 Pong Game", CKgPoint(0, 0));
  DrawSceneTextPos("Life: ", CKgPoint(250, 0));
  DrawSceneTextPos("Score: ", CKgPoint(400, 0));
  std::string score_string = std::to_string(score);
  std::string life_string = std::to_string(life);
  DrawSceneTextPos(life_string.c_str(), CKgPoint(320, 0));
  DrawSceneTextPos(score_string.c_str(), CKgPoint(460, 0));
```

게임의 기본적인 룰 설정 (클리어 및 종료)

```
/*목숨이 0가 되면 게임 종료*/
  if (life <= 0)
    DrawSceneTextPos("Game Over, Press ESC to exit", CKgPoint(180, 270));

  /*스코어가 240이면 모든 벽돌을 깼다는 것임->볼을 정지함.*/
  if (score >= 240)
  {
    DrawSceneTextPos("Game Clear, Press ESC to exit", CKgPoint(180, 270));
    //m_pScene->m_Children
    for (auto& Layer : m_pScene->m_Children) {
      for (auto& Sprite : Layer->m_Children) {
        CKhuGleSprite* Ball = static_cast<CKhuGleSprite*>(Sprite);
        if (Ball->m_nType == GP_STYPE_ELLIPSE)
          Ball->m_Velocity = CKgVector2D(0, 0);
      }
    }
  }
```