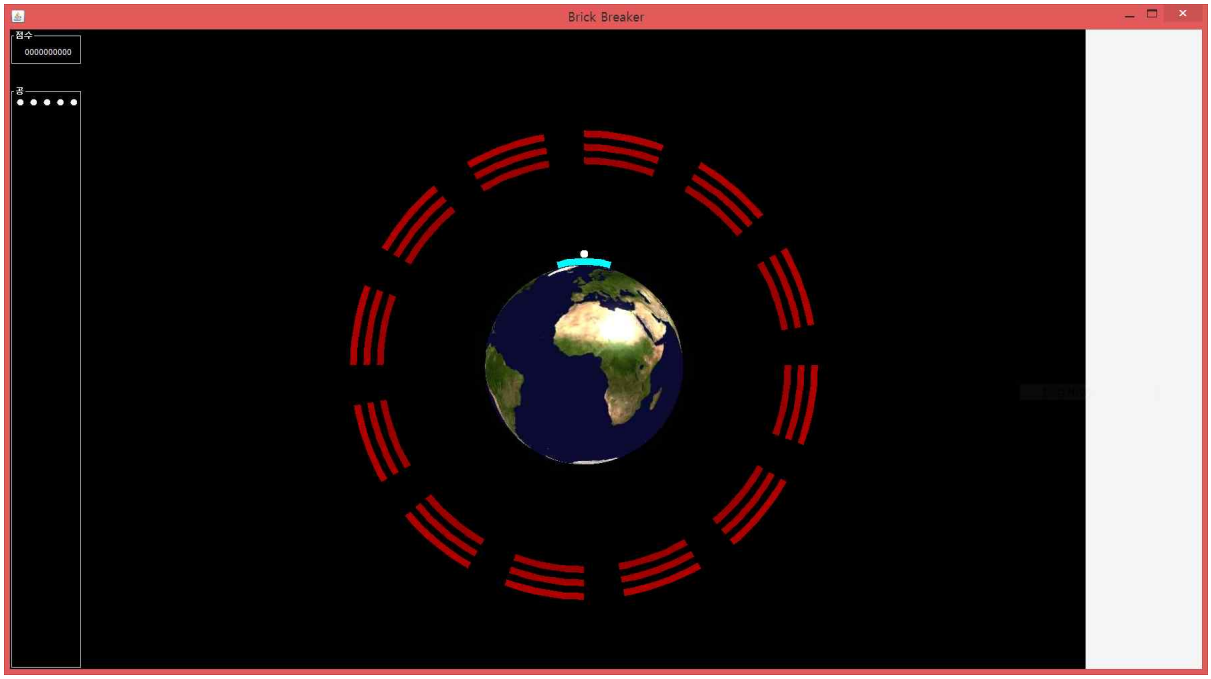
16.5.5~10
주제를 ‘행성에서 벽돌깨기’로 정하고 코딩을 했다.
‘행성에서 벽돌깨기’는 기존의 벽돌깨기를 행성에서 한다고 보면 된다. 조금 더 정확히 말하자면, 행성 주변을 벽돌들이 둘러싸고 있고, 이 벽돌들이 행성에 떨어져 닿기 전에 공으로 벽돌들을 모두 깨는 게임이다.
아래는 실행 화면이다. 설명이 이해가 가지 않는다면 아래 그림을 보는 편이 좋다.



빨간색이 벽돌이고, 벽돌은 높이(지구로부터의 거리)에 따라서 색이 점점 짙어지도록 했다.
벽돌과 비슷하게 생긴 cyan색이 공을 튕기는 판이다. 키보드의 왼쪽 방향키를 누르면 판이 시계 반대 방향으로 돌고, 오른쪽 방향키를 누르면 판이 시계 방향으로 돌도록 했다.

프로그램의 설명은 보고서에서 조금 더 자세하게 해야겠다.

2016.06.13
해야 할 것 : 아이템 추가, Thread 사용, 스테이지 추가, 정지 화면과 메인 화면 추가, 그래픽 요소 추가

1.보드 길이 연장, 공 고정(?), 공 추가 등
2.Thread를 어디에 써야 할지 모르겠다.

3.스테이지는 태양계+명왕성으로 11개

4.정지 화면은 PausePanel이라는 클래스를 만든 후, gamepanel과 JLayerPane을 이용해 위치를 바꾸는 식으로 하려 했는데 잘 안된다.(역시 Java…)

메인 화면은 만들어야 하는데 PausePanel도 제대로 안되는 관계로 일단 보류

5.행성이 회전하는 것을 구현할 생각이다. 되도록이면 자전축의 기울어진 각도도 고려하고 싶은데 아직 잘 안된다. 수직축을 기준으로 자전이나 공전시키는 것은 아래 코드를 잘 만지면 된다.(빡세다ㄷㄷ)

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GraphicsConfigTemplate;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.image.BufferedImage;
import java.awt.image.RenderedImage;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import javax.imageio.IIOImage;
import javax.imageio.ImageIO;
import javax.imageio.ImageTypeSpecifier;
import javax.imageio.ImageWriter;
import javax.imageio.metadata.IIOMetadata;
import javax.imageio.stream.ImageOutputStream;
import javax.media.j3d.Alpha;
import javax.media.j3d.AmbientLight;
import javax.media.j3d.Appearance;
import javax.media.j3d.Background;
import javax.media.j3d.BoundingSphere;
import javax.media.j3d.Bounds;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Canvas3D;
import javax.media.j3d.DirectionalLight;
import javax.media.j3d.GraphicsConfigTemplate3D;
import javax.media.j3d.Group;
import javax.media.j3d.ImageComponent;
import javax.media.j3d.ImageComponent2D;
import javax.media.j3d.Locale;
import javax.media.j3d.Material;
import javax.media.j3d.PhysicalBody;
import javax.media.j3d.PhysicalEnvironment;
import javax.media.j3d.PointLight;
import javax.media.j3d.RotationInterpolator;
```

```
import javax.media.j3d.Texture;
import javax.media.j3d.TextureAttributes;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import javax.media.j3d.View;
import javax.media.j3d.ViewPlatform;
import javax.media.j3d.VirtualUniverse;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.WindowConstants;
import javax.vecmath.Color3f;
import javax.vecmath.Point3d;
import javax.vecmath.Vector3d;
import javax.vecmath.Vector3f;
import com.sun.j3d.utils.geometry.Primitive;
import com.sun.j3d.utils.geometry.Sphere;
import com.sun.j3d.utils.image.TextureLoader;
public class Test2d extends JPanel {
 private BranchGroup sceneBranchGroup = null;
 private RotationInterpolator    rotator = null;
 private static Canvas3D       offScreenCanvas3D = null;
 private static ImageComponent2D      imageComponent = null;
 private static final int    offScreenWidth = 400;
 private static final int    offScreenHeight = 400;
 public Test2d()
 {
  setLayout( new BorderLayout() );
  init();
 }
 protected void init()
 {
  VirtualUniverse universe = createVirtualUniverse();
  Locale locale = createLocale( universe );
  BranchGroup sceneBranchGroup = createSceneBranchGroup();
  Background background = createBackground();
  if( background != null )
   sceneBranchGroup.addChild( background );
  ViewPlatform vp = createViewPlatform();
  BranchGroup viewBranchGroup =
     createViewBranchGroup( getViewTransformGroupArray(), vp );
  locale.addBranchGraph( sceneBranchGroup );
  addViewBranchGroup( locale, viewBranchGroup );
  createView( vp );
 }
 protected void addCanvas3D( Canvas3D c3d )
 {
  add( "Center", c3d );
 }
```

```
protected View createView( ViewPlatform vp )
{
  View view = new View();
  PhysicalBody pb = createPhysicalBody();
  PhysicalEnvironment pe = createPhysicalEnvironment();
  view.setPhysicalEnvironment( pe );
  view.setPhysicalBody( pb );
  if( vp != null )
    view.attachViewPlatform( vp );
  view.setBackClipDistance( getBackClipDistance() );
  view.setFrontClipDistance( getFrontClipDistance() );
  Canvas3D c3d = createCanvas3D( false );
  view.addCanvas3D( c3d );
  view.addCanvas3D( createOffscreenCanvas3D() );
  addCanvas3D( c3d );
  return view;
}

protected Background createBackground()
{
  Background back = new Background(new Color3f( 0.0f, 0.0f, 0.0f ) );
  back.setApplicationBounds( createApplicationBounds() );
  return back;
}

protected Bounds createApplicationBounds()
{
  return new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
}

protected Canvas3D createCanvas3D( boolean offscreen )
{
  GraphicsConfigTemplate3D gc3D = new GraphicsConfigTemplate3D();
  gc3D.setSceneAntialiasing( GraphicsConfigTemplate.PREFERRED );
  GraphicsDevice gd[] = GraphicsEnvironment.
      getLocalGraphicsEnvironment().
      getScreenDevices();
  Canvas3D c3d = new Canvas3D( gd[0].getBestConfiguration( gc3D ), offscreen );
  c3d.setSize( 500, 500 );
  return c3d;
}

protected double getScale()
{
  return 3;
}
public TransformGroup[] getViewTransformGroupArray()
{
```

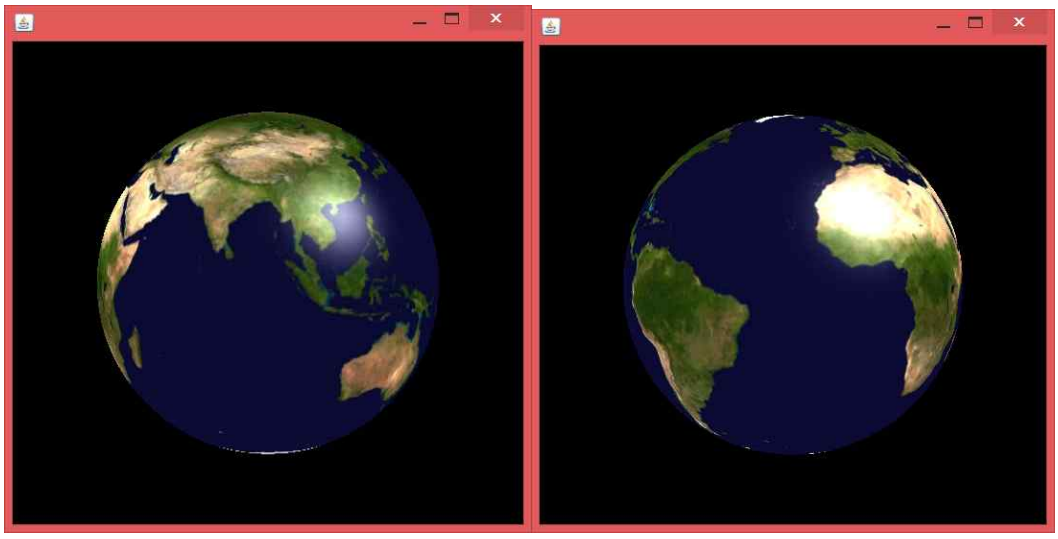
```
  TransformGroup[] tgArray = new TransformGroup[1];
  tgArray[0] = new TransformGroup();
  Transform3D t3d = new Transform3D();
  t3d.setScale( getScale() );
  t3d.setTranslation( new Vector3d( 0.0, 0.0, -40.0 ) );
  t3d.invert();
  tgArray[0].setTransform( t3d );
  return tgArray;
}
protected void addViewBranchGroup( Locale locale, BranchGroup bg )
{
  locale.addBranchGraph( bg );
}
protected Locale createLocale( VirtualUniverse u )
{
  return new Locale( u );
}
protected PhysicalBody createPhysicalBody()
{
  return new PhysicalBody();
}
protected PhysicalEnvironment createPhysicalEnvironment()
{
  return new PhysicalEnvironment();
}
protected float getViewPlatformActivationRadius()
{
  return 100;
}
protected ViewPlatform createViewPlatform()
{
  ViewPlatform vp = new ViewPlatform();
  vp.setViewAttachPolicy( View.RELATIVE_TO_FIELD_OF_VIEW );
  vp.setActivationRadius( getViewPlatformActivationRadius() );
  return vp;
}
protected double getBackClipDistance()
{
  return 100.0;
}
protected double getFrontClipDistance()
{
  return 1.0;
}
protected BranchGroup createViewBranchGroup(
    TransformGroup[] tgArray, ViewPlatform vp )
{
  BranchGroup vpBranchGroup = new BranchGroup();
```

```
  if( tgArray != null && tgArray.length > 0 )
  {
    Group parentGroup = vpBranchGroup;
    TransformGroup curTg = null;
    for( int n = 0; n < tgArray.length; n++ )
    {
      curTg = tgArray[n];
      parentGroup.addChild( curTg );
      parentGroup = curTg;
    }
    tgArray[tgArray.length-1].addChild( vp );
  }
  else
    vpBranchGroup.addChild( vp );
  return vpBranchGroup;
}
protected VirtualUniverse createVirtualUniverse()
{
  return new VirtualUniverse();
}
protected Canvas3D createOffscreenCanvas3D()
{
  offScreenCanvas3D = createCanvas3D( true );
  offScreenCanvas3D.getScreen3D().setSize( offScreenWidth,
     offScreenHeight );
  offScreenCanvas3D.getScreen3D().
  setPhysicalScreenHeight( 0.0254/90 * offScreenHeight );
  offScreenCanvas3D.getScreen3D().
  setPhysicalScreenWidth( 0.0254/90 * offScreenWidth );
  RenderedImage renderedImage =
     new BufferedImage( offScreenWidth, offScreenHeight,
        BufferedImage.TYPE_3BYTE_BGR );
  imageComponent =
     new ImageComponent2D( ImageComponent.FORMAT_RGB8,
        renderedImage );
  imageComponent.setCapability( ImageComponent2D.ALLOW_IMAGE_READ );
  //Finally, we assign the ImageComponent2D to the offscreen
  //Canvas3D for rendering
  offScreenCanvas3D.setOffScreenBuffer( imageComponent );
  return offScreenCanvas3D;
}

protected BranchGroup createSceneBranchGroup()
{
  BranchGroup objRoot = new BranchGroup();
  TransformGroup objTrans = new TransformGroup();
  objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
  objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
```

```
  BoundingSphere bounds = new BoundingSphere(
     new Point3d(0.0,0.0,0.0), 100.0);
  Transform3D yAxis = new Transform3D();

  Alpha rotationAlpha = new Alpha(-1, Alpha.INCREASING_ENABLE,
     0, 0,
     4000, 0, 0,
     0, 0, 0);
  rotator = new RotationInterpolator( rotationAlpha,
     objTrans, yAxis, 0.0f,
     (float) Math.PI*2.0f );

  rotator.setSchedulingBounds( bounds );
  objTrans.addChild(rotator);
  sceneBranchGroup = new BranchGroup();
  sceneBranchGroup.setCapability( Group.ALLOW_CHILDREN_EXTEND );
  sceneBranchGroup.setCapability( Group.ALLOW_CHILDREN_READ );
  sceneBranchGroup.setCapability( Group.ALLOW_CHILDREN_WRITE );
  sceneBranchGroup.addChild( createSphere() );
  Color3f lColor1 = new Color3f( 0.7f,0.7f,0.7f );
  Vector3f lDir1    = new Vector3f( -1.0f,-1.0f,-1.0f );
  Color3f alColor = new Color3f( 0.2f,0.2f,0.2f );
  AmbientLight aLgt = new AmbientLight( alColor );
  aLgt.setInfluencingBounds( bounds );
  //Create a directional light
  DirectionalLight lgt1 = new DirectionalLight( lColor1, lDir1 );
  lgt1.setInfluencingBounds( bounds );
  //Add the lights to the scenegraph
  objRoot.addChild(aLgt);
  objRoot.addChild(lgt1);
  objTrans.addChild( sceneBranchGroup );
  objRoot.addChild( objTrans );
  //Return the root of the scene side of the scenegraph
  return objRoot;
}

protected BranchGroup createSphere()
{
  BranchGroup group = new BranchGroup();
  group.addChild(basicSphere(0,0,0));

  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 10000000.0);
  PointLight light = new PointLight();
  light.setColor(new Color3f(Color.WHITE));
  light.setPosition(0.0f,0.0f,0.0f);
  light.setInfluencingBounds(bounds);
  group.addChild(light);
  return group;
```

```
 }
 private TransformGroup basicSphere(double x, double y, double z) {
  try {
   int primflags = Primitive.GENERATE_NORMALS +
Primitive.GENERATE_TEXTURE_COORDS;
   Texture tex = new TextureLoader(
     ImageIO.read(new FileInputStream("Planet/earth.jpg"))
     ).getTexture();
   tex.setBoundaryModeS(Texture.WRAP);
   tex.setBoundaryModeT(Texture.WRAP);
   TextureAttributes texAttr = new TextureAttributes();
   Appearance ap = new Appearance();
   ap.setTexture(tex);
   ap.setTextureAttributes(texAttr);
   Material material = new Material();
   material.setSpecularColor(new Color3f(Color.WHITE));
   material.setDiffuseColor(new Color3f(Color.WHITE));
   ap.setMaterial(material);
   Sphere sphere = new Sphere(2.0f, primflags, 100, ap);
   Transform3D transform = new Transform3D();
   transform.setTranslation(new Vector3d(0.0f,0.0f,0.0f));
   TransformGroup transformGroup = new TransformGroup();
   transformGroup.setTransform(transform);
   transformGroup.addChild(sphere);
   return transformGroup;
  } catch(Exception e) { e.printStackTrace(); }
  return null;
 }
 protected static void onSaveImage()
 {
  offScreenCanvas3D.renderOffScreenBuffer();
  offScreenCanvas3D.waitForOffScreenRendering();
  try
  {
   FileOutputStream fileOut = new FileOutputStream( "Planet/image.jpg" );
   BufferedImage challenge=imageComponent.getImage();
   ImageWriter imageWriter =
(ImageWriter)ImageIO.getImageWritersBySuffix("jpeg").next();
   ImageOutputStream ios = ImageIO.createImageOutputStream(fileOut);
   imageWriter.setOutput(ios);
   IIOMetadata imageMetaData = imageWriter.getDefaultImageMetadata(new
ImageTypeSpecifier(challenge), null);
   imageWriter.write(imageMetaData, new IIOImage(challenge, null, null), null);
  }
  catch( Exception e )
  {
  }
 }
```

```
 static protected void registerWindowListener( JFrame frame )
 {
  frame.setDefaultCloseOperation( WindowConstants.DO_NOTHING_ON_CLOSE );
  frame.addWindowListener(
     new WindowAdapter()
     {
      public void windowClosing( WindowEvent e )
      {
       onSaveImage();
       System.exit( 1 );
      }
     }
     );
 }
 public static void main( String[] args )
 {
  JFrame frame = new JFrame();
  Test2d swingTest = new Test2d();

  frame.getContentPane().add( swingTest );
  frame.setSize( 550, 550 );
  registerWindowListener( frame );
  frame.setVisible( true );
 }
}
```



y축을 기준으로 잘 자전한다.

2016.06.15
-스테이지 11개를 모두 완성했다. 각 스테이지마다 행성의 크기, 벽돌의 배치 등이 다르게 했다.

-저 위에 있는 코드를 이용해 gamepanel의 배경에서 행성이 자전하는 것처럼 보이도록 하고 싶어서 무작정 매번 onSaveImage()를 했더니 시간도 엄청 오래 걸리고 이미지를 불러올 때 이미지가 수정되는 중이어서 배경이 그냥 회색으로 나타나는 등 여러 문제가 있었다.
->Thread를 사용하자!!
=>Thread를 사용해서 위 같은 문제가 발생해서 일단 보류

Thread를 어디에 쓰지????????
->timer1,2를 모두 Thread로 바꿈
=>Dialog가 안 닫아짐, 그래도 그냥 둠.

-메인 화면이랑 PausePanel : 아직 만들지 않는다.

-위에서 thread를 이용해서 행성이 자전하는 것처럼 보이게 하는 것에 실패했으나, 일단 임의의 각도로 기울어져서 회전하도록 해 봤는데 그것도 안됨 (stack overflow에서 된다는데 분명...)

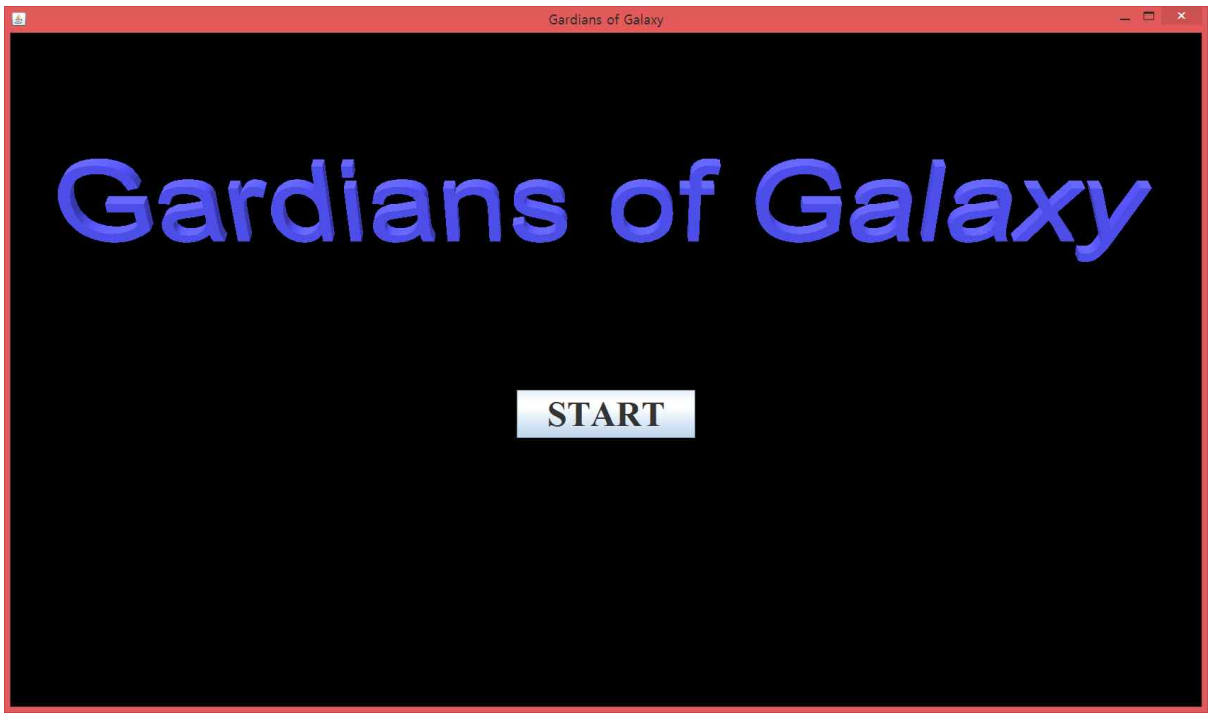
2016.06.17
-Thread를 Timer 대신 사용하는데 성공
-MyDialog가 이제 작동한다 -> 다시 시도나 다음 단계로 넘어가는 것이 가능해짐

2016.06.18
-아이템을 추가했다. 공 추가, 보드 길이 연장, 보드 자석(?) 등등
-원래는 아이템이 떨어지면 먹었을 때 효과가 나타나야 하는데, 공을 튕기면서 아이템까지 먹기가 힘들다는 몇몇 친구들의 의견이 있어서 다시 열심히 바꿨다.
-얼떨결에 몇몇 아이템의 효과를 위해서 Thread를 사용하게 되었다(Thread가 편하다는 점을 오늘 처음 느꼈다.)
-뭔가 아이템을 더 넣고 싶은데 아이디어가 모자라다.(아주 만약에 나중에 요청이 들어오면(?) 고치도록 하자)

2016.06.19

-Java3D를 사용해 메인 화면을 대충 만들었다.

게임 제목을 ‘Gardians of Earth’에서 ‘Gardians of Galaxy’(feat.Marvel)로 바꾸었다.



-자잘한 오류 처리

-난이도 보정