TAGE Physics

```
. . .
import tage.physics.PhysicsEngine;
import tage.physics.PhysicsObject;
import tage.physics.PhysicsEngineFactory;
import tage.physics.JBullet.*;
import com.bulletphysics.dynamics.RigidBody;
import com.bulletphysics.collision.dispatch.CollisionObject;

public class MyGame extends VariableFrameRateGame
{  . . .
   private GameObject ball1, ball2, plane;
   private ObjShape sphS, planeS;
   private TextureImage ice, brick;

   private PhysicsEngine physicsEngine;
   private PhysicsObject ball1P, ball2P, planeP;

   private boolean running = false;
   private float vals[] = new float[16];
   . . .
```

GRAPHICS WORLD

```
   @Override
   public void buildObjects()
   {  // ------------- adding two Spheres ----------------
      ball1 = new GameObject(GameObject.root(), sphS, ice);
      ball1.setLocalTranslation((new Matrix4f()).translation(0, 4, 0));
      ball1.setLocalScale((new Matrix4f()).scaling(0.75f));

      ball2 = new GameObject(GameObject.root(), sphS, ice);
      ball2.setLocalTranslation((new Matrix4f()).translation(-0.5f, 1, 0));
      ball2.setLocalScale((new Matrix4f()).scaling(0.75f));

      // ------------- adding a ground plane ------------
      plane = new GameObject(GameObject.root(), planeS, brick);
      plane.setLocalTranslation((new Matrix4f()).translation(0, -2.75f, 0));
      plane.setLocalScale((new Matrix4f()).scaling(4.0f));
   }

   @Override
   public void update()
   {  Matrix4f currentTranslation, currentRotation;
      double totalTime = System.currentTimeMillis() - startTime;
      elapsedTime = System.currentTimeMillis() - prevTime;
      prevTime = System.currentTimeMillis();
      amt = elapsedTime * 0.03;
      double amtt = totalTime * 0.001;

      // update physics
      if (running)
      {  Matrix4f mat = new Matrix4f();
         Matrix4f mat2 = new Matrix4f().identity();
         checkForCollisions();
         physicsEngine.update((float)elapsedTime);
         for (GameObject go:engine.getSceneGraph().getGameObjects())
         {  if (go.getPhysicsObject() != null)
            {  mat.set(toFloatArray(go.getPhysicsObject().getTransform()));
               mat2.set(3,0,mat.m30());
               mat2.set(3,1,mat.m31());
               mat2.set(3,2,mat.m32());
               go.setLocalTranslation(mat2);
   } } } }
```

PHYSICS WORLD

```
   public void initializeGame()
   {  . . .
      //    --- initialize physics system ---
      String engine = "tage.physics.JBullet.JBulletPhysicsEngine";
      float[] gravity = {0f, -5f, 0f};
      physicsEngine = PhysicsEngineFactory.createPhysicsEngine(engine);
      physicsEngine.initSystem();
      physicsEngine.setGravity(gravity);

      //    --- create physics world ---
      float mass = 1.0f;
      float up[ ] = {0,1,0};
      double[ ] tempTransform;

      Matrix4f translation = new Matrix4f(ball1.getLocalTranslation());
      tempTransform = toDoubleArray(translation.get(vals));
      ball1P = physicsEngine.addSphereObject(physicsEngine.nextUID(),
                                             mass, tempTransform, 0.75f);
      ball1P.setBounciness(1.0f);
      ball1.setPhysicsObject(ball1P);

      translation = new Matrix4f(ball2.getLocalTranslation());
      tempTransform = toDoubleArray(translation.get(vals));
      ball2P = physicsEngine.addSphereObject(physicsEngine.nextUID(),
                                             mass, tempTransform, 0.75f);
      ball2P.setBounciness(1.0f);
      ball2.setPhysicsObject(ball2P);

      translation = new Matrix4f(plane.getLocalTranslation());
      tempTransform = toDoubleArray(translation.get(vals));
      planeP = physicsEngine.addStaticPlaneObject(
                  physicsEngine.nextUID(), tempTransform, up, 0.0f);
      planeP.setBounciness(1.0f);
      plane.setPhysicsObject(planeP);
   }

   private void checkForCollisions()
   {  com.bulletphysics.dynamics.DynamicsWorld dynamicsWorld;
      com.bulletphysics.collision.broadphase.Dispatcher dispatcher;
      com.bulletphysics.collision.narrowphase.PersistentManifold manifold;
      com.bulletphysics.dynamics.RigidBody object1, object2;
      com.bulletphysics.collision.narrowphase.ManifoldPoint contactPoint;

      dynamicsWorld =
            ((JBulletPhysicsEngine)physicsEngine).getDynamicsWorld();
      dispatcher = dynamicsWorld.getDispatcher();
      int manifoldCount = dispatcher.getNumManifolds();
      for (int i=0; i<manifoldCount; i++)
      {  manifold = dispatcher.getManifoldByIndexInternal(i);
         object1 =
            (com.bulletphysics.dynamics.RigidBody)manifold.getBody0();
         object2 =
            (com.bulletphysics.dynamics.RigidBody)manifold.getBody1();
         JBulletPhysicsObject obj1 =
            JBulletPhysicsObject.getJBulletPhysicsObject(object1);
         JBulletPhysicsObject obj2 =
            JBulletPhysicsObject.getJBulletPhysicsObject(object2);
         for (int j = 0; j < manifold.getNumContacts(); j++)
         {  contactPoint = manifold.getContactPoint(j);
            if (contactPoint.getDistance() < 0.0f)
            {  System.out.println("---- hit between " + obj1 + " and " + obj2);
               break;
   } } } }

   public void keyPressed(KeyEvent e)
   {  // map space bar to setting the variable "running" to TRUE
      . . .
   }
```

```
// ------------------ UTILITY FUNCTIONS used by physics

private float[] toFloatArray(double[] arr)
{   if (arr == null) return null;
    int n = arr.length;
    float[] ret = new float[n];
    for (int i = 0; i < n; i++)
    {   ret[i] = (float)arr[i];
    }
    return ret;
}

private double[] toDoubleArray(float[] arr)
{   if (arr == null) return null;
    int n = arr.length;
    double[] ret = new double[n];
    for (int i = 0; i < n; i++)
    {   ret[i] = (double)arr[i];
    }
    return ret;
}
```