

User Interaction

Input Handling

Cameras

Picking

Separating application logic

Input Handling

InputManager

- Field of SimpleApplication
- Handles input from
 - Mouse
 - Keyboard
 - Joystick
- Is event driven, listen and take action accordingly

Input Triggers

- Input triggers
 - Key press or mouse action
 - `new KeyTrigger(KeyInput.KEY_P)`
- Different types of triggers
 - KeyTrigger
 - MouseAxisTrigger
 - MouseButtonTrigger
 - JoyAxisTrigger
 - JoyButtonTrigger

Input Mappings

- Input Mappings
 - String name (case sensitive)
 - Maps to one or more triggers

```
inputManager.addMapping("Pause Game",  
    new KeyTrigger(KeyInput.KEY_P));  
inputManager.addMapping(MoveUpDown,  
    new MouseAxisTrigger(MouseInput.AXIS_Y, true),  
    new MouseAxisTrigger(MouseInput.AXIS_Y, false));
```

InputManager

- Add input listeners to handle input

```
inputManager.addListener(actionListener, "Pause Game");
```

- Listener types:

- ActionListener (on/off)

```
public void onAction(String name, boolean keyPressed, float tpf)
```

- AnalogListener (continious)

```
public void onAnalog(String name, float value, float tpf)
```

- TouchListener (touch devices)

```
Public void onTouch(String name, TouchEvent evt, float tpf)
```

- Callback to listener contains only the named input mapping

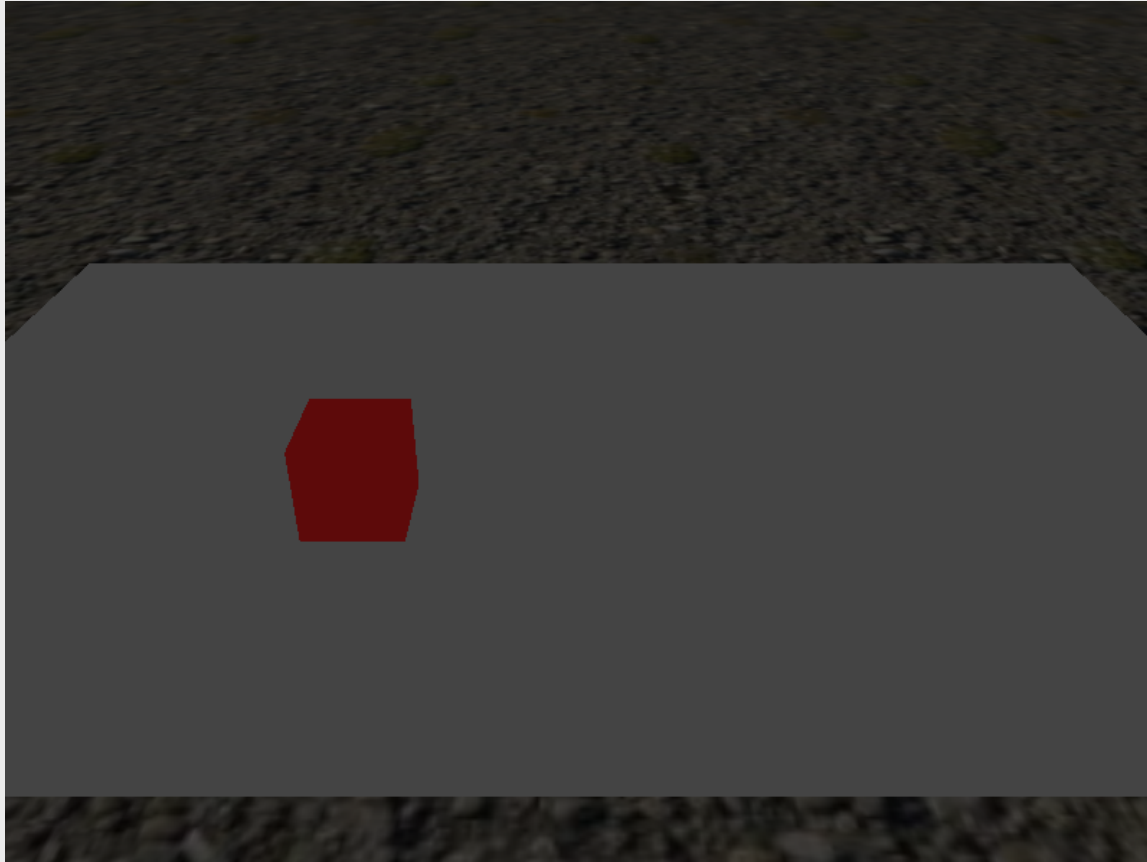
Input Example



InputExample.java

TWi March 15

Navigation example



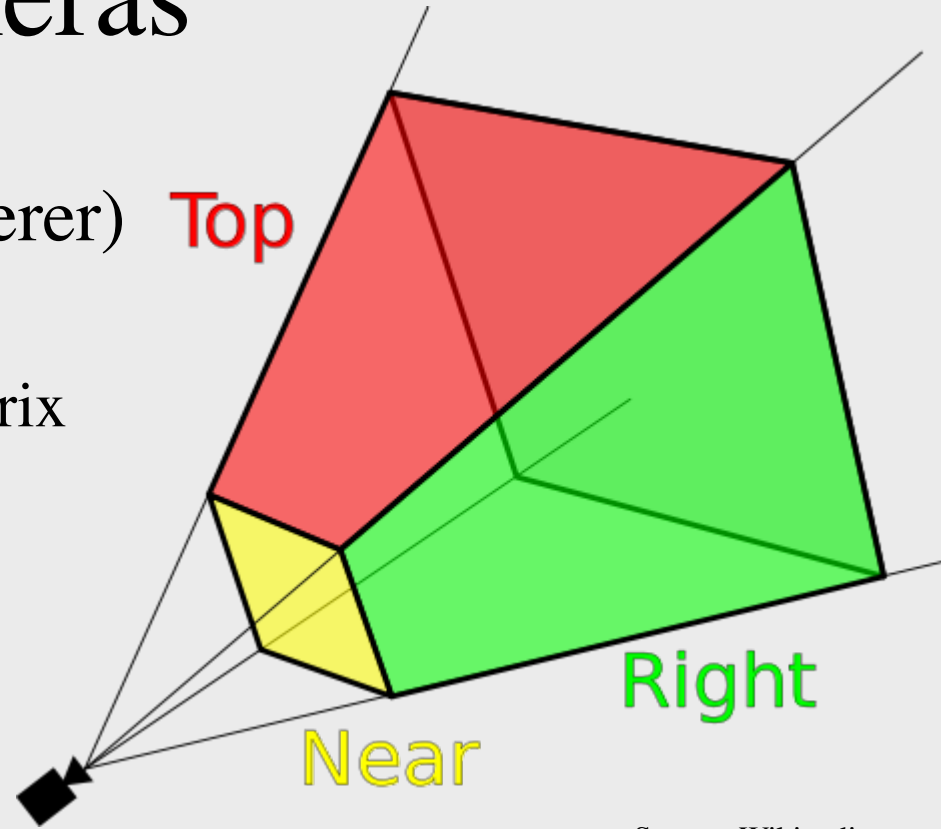
NavigationExample.java

TWi March 15

Cameras

Cameras

- Camera (com.jme3.renderer)
 - Purely mathematical
 - View and projection matrix
 - Frustrum
 - Location and rotation
 - Forward, up, and right direction vectors
 - Used by camera implementations
 - Culling

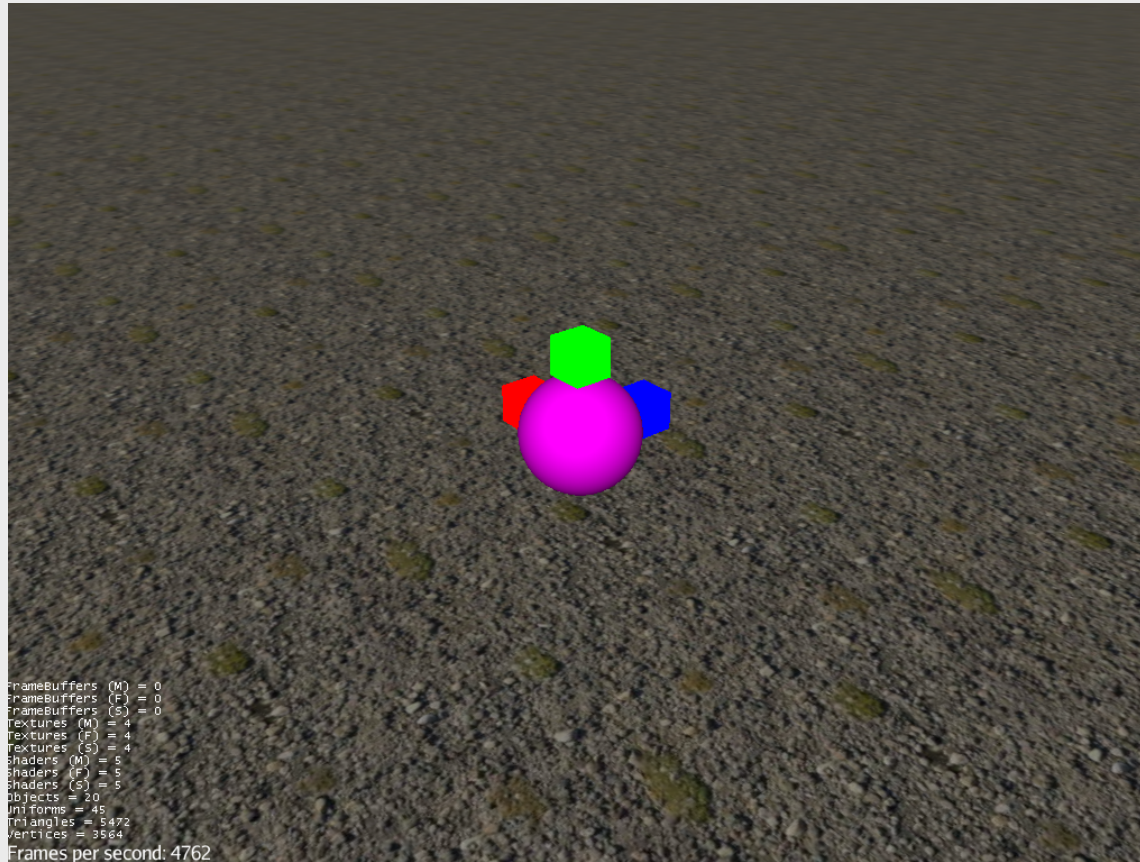


Source: Wikipedia

Cameras

- Various camera implementations:
 - FlyByCamera
 - First person controls
 - ChaseCamera
 - Third person controls, follows with a smooth transition
 - CameraNode
 - Third person, fixed distance
 - (ExamineCamera)

Camera Example



CameraTypesExample.java

TWi March 15

Picking

Picking

- Ray Casting
- Intersection with Bounding Volumes
 - Axis Aligned Bounding Box (AABB)
 - BoundingSphere
 - Oriented Bounding Box (OBB)
 - Capsule
- If collision with BV, per triangle intersection

Picking

- Construct a ray with a from location and a direction

```
Ray ray = new Ray(Vector3f.ZERO, Vector3f.NEGATIVE_Z);
```

- CollisionResults stores the result from the pick operation

```
CollisionResults results = new CollisionResults();
```

- Check collision with subgraph

```
subgraphToPick.collideWith(ray, results);
```

- Get the collision

```
CollisionResult closest = results.getClosestCollision();  
results.getFarthestCollision();
```

```
public Iterator<CollisionResult> iterator()
```

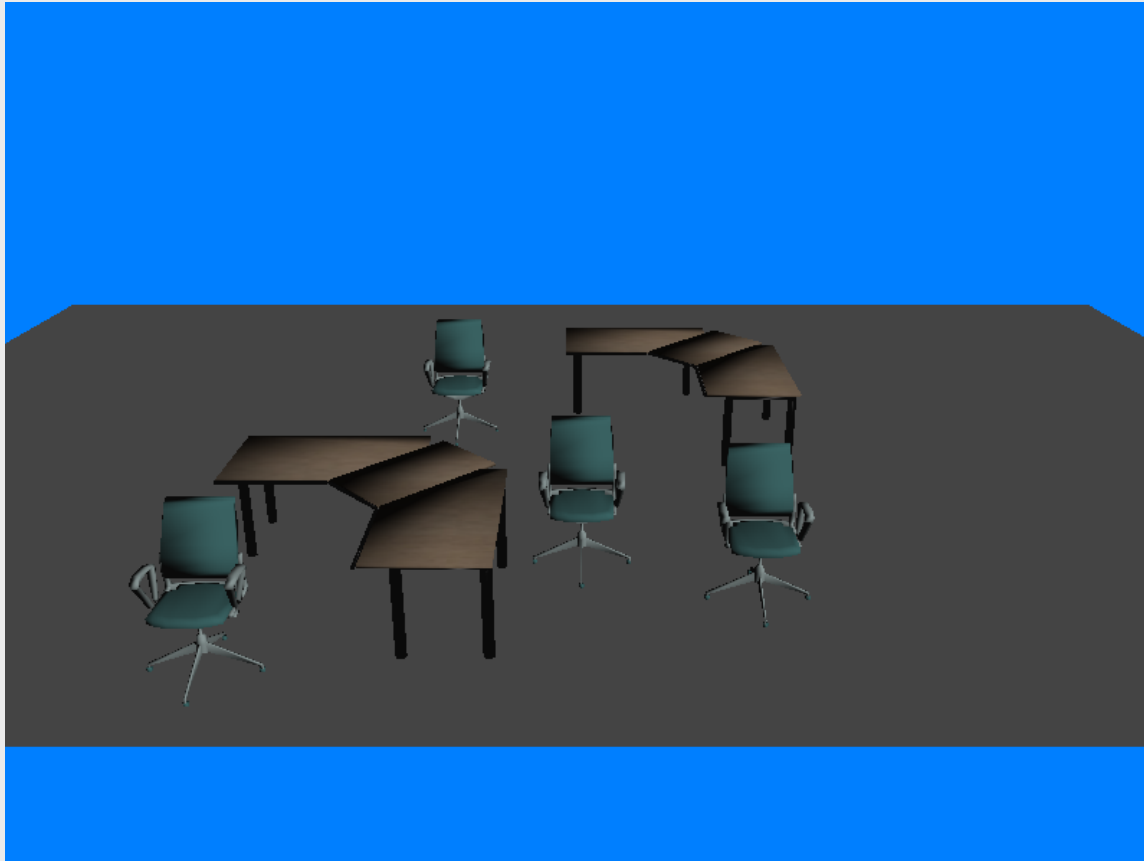
Picking

- The pick result contains detailed information
 - Geometry, mesh and triangle
 - Point, normal and distance
- Results can be sorted
- Note: jME counts intersection with front and back of a mesh as two hits

Picking

- Pickable/Collidable objects must implement Collidable interface
 - Spatial, Node and Geometry
- BV can be used to check collision between shapes
 - Much cheaper than physics collision (simulation)

Picking Example



PickingExample.java

TWi March 15

Controls

Controls

- Control contains code/behavior specific to individual Spatial or types of spatial
- Scope of a Control is limited to the Spatial (and its subgraph)
- One Spatial can be influenced by several Controls
- Each Spatial needs its own instance of the Control
- Controls can be saved in the .j3o file together with a Spatial.

Controls

- Each Control has:
 - Constructor, cannot modify the spatial here
 - A `setSpatial(Spatial ..)` method, where you can do initial modifications to the spatial
 - Its own `update()` loop that hooks into `simpleUpdate()`
 - Access to other controls added to the spatial
- Controls move blocks of code out of the `simpleUpdate()` loop
- Create a control by either extending `AbstractControl` or implement `Control` interface

Custom Control Example

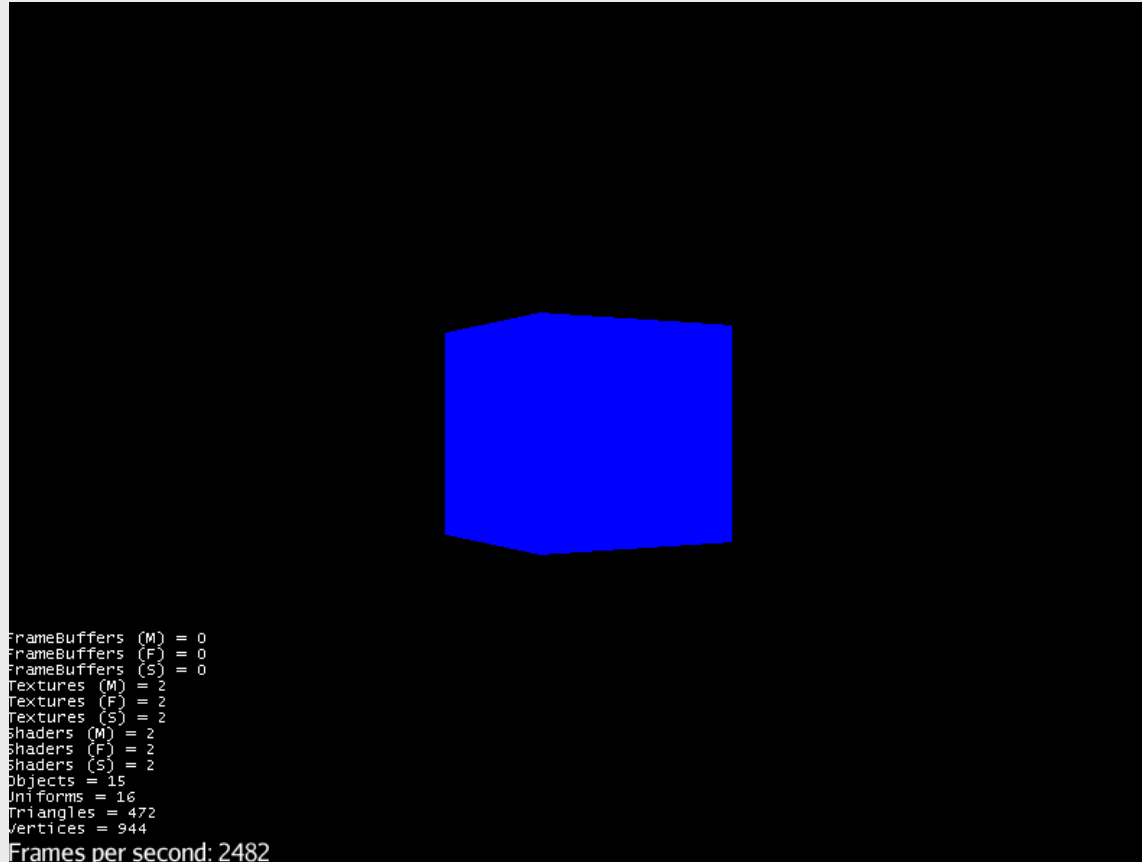
```
Public class MyControl extends AbstractControl{
    public MyControl(Params...){
    }

    public void setSpatial(Spatial spatial){
        super.setSpatial(spatial);
        // control specification initializing code here
    }

    public void controlUpdate(float tpf){
        // update code here
    }

    public Control cloneForSpatial(Spatial spatial){ ... }
    public void controlRender(RenderManager rm, ViewPort vp){ ... }
    public void read(JmeImporter im) throws IOException{ ... }
    public void write(JmeExporter ex) throws IOException{ ... }
}
```

Simple Control Example



RotationWithControl.java

TWi March 15

Application States

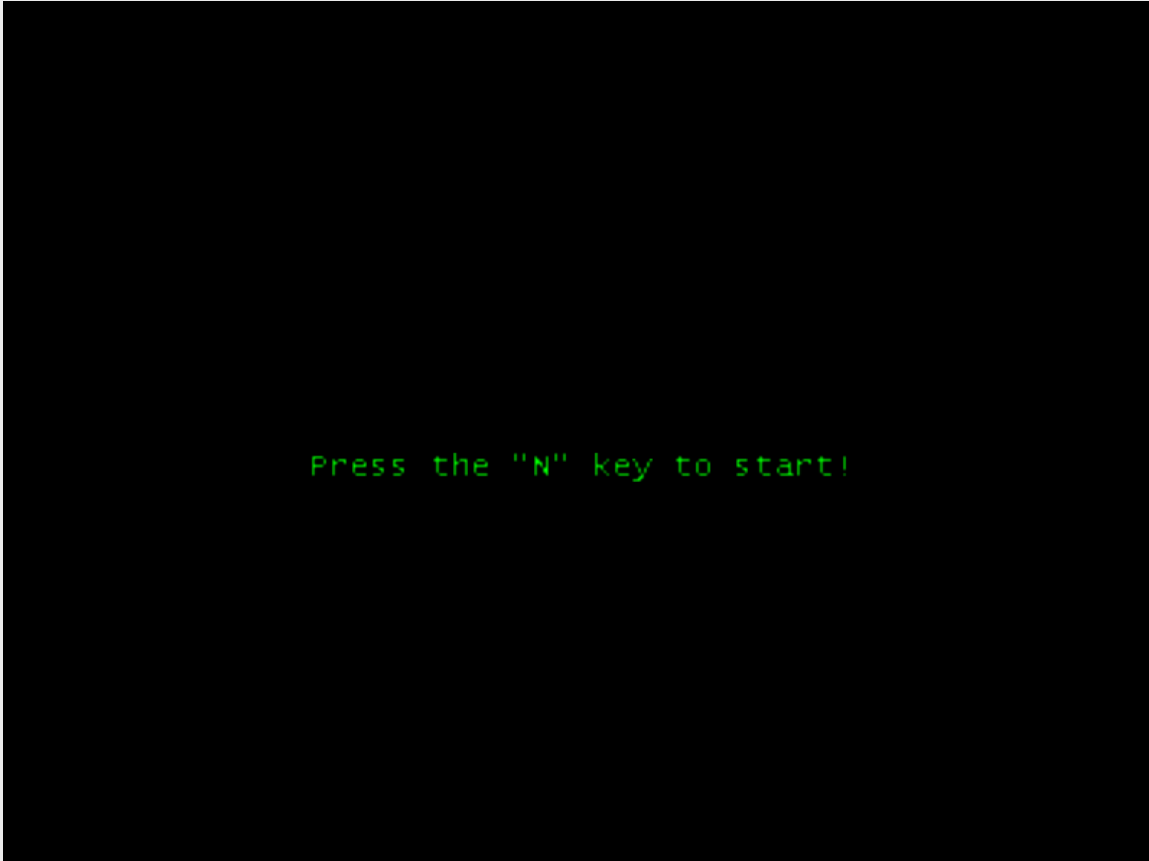
Application States

- Separation of game logic
- Where Controls enabled logic specific to spatial
- Application States enable logic specific to parts of the application / game
- Application States have access to the whole Application

Application States

- Application State has various methods:
 - initialize(AppStateManager stateManager, Application app)
 - setEnabled(boolean enabled)
 - stateAttached(AppStateManager stateManager)
 - stateDetached(AppStateManager stateManager)
 - update(float tpf)
 - postRender()
 - cleanup()
 - ++

Application State Example



Press the "N" key to start!

AppStateExample.java

TWi March 15