cisc3660
game programming
fall 2012
lecture # II.1

topics:

- event handling

references:

- `https://sites.google.com/site/blendergameprojects/`,
  by Prof Tim Hickey, Brandeis University (`http://www.cs.brandeis.edu/~tim`)
- Blender Game Engine Overview, User Manual version 2.6
  `http://wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine`

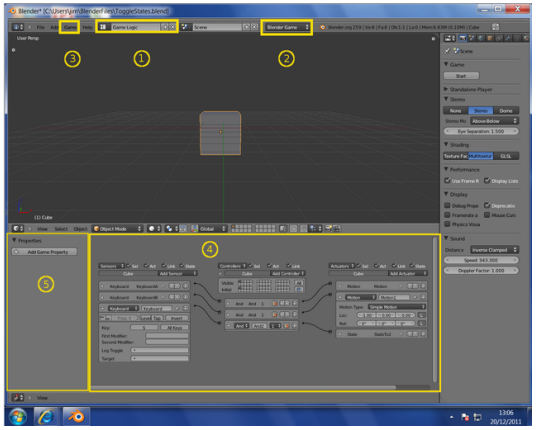
event handling

- **events** can be caused by the user in an interactive environment
- **events** can also be caused by activity in a virtual world
- an **event handler** is the code that gets executed when an **event** is detected
- recall that when we learned about events in JavaScript, we had to do two things:
  - define an event handler
    in JavaScript, this is a function defintion like this:
    `function FUNCTION_NAME ( ... ) { ... }`
  - associate the event handler with the object in the interface/environment, so that the event handler is triggered when the registered event occurs
    in JavaScript, this is a call like this:
    `OBJECT.addEventListener( EVENT, FUNCTION_NAME, ... )`
- today we'll talk about handling events in Blender

blender game engine

- the **Game Engine** aspect of Blender includes the **Physics** simulator (that we discussed in the last class) and a **Logic Editor** that lets us define how the game behaves
- four steps to creating a game:
  1. create visual elements (3D models or images, could be rendered objects)
  2. use logic editor to create behaviors for the game
     can define how user interacts with objects
     and how objects interact with each other
     and how objects interact with the environment
  3. create camera(s) from which to render the scene
  4. launch the game (e.g., create a runtime version)
- compnents of game logic
  - logic "bricks"
  - properties (like variables)
  - states (an object property)

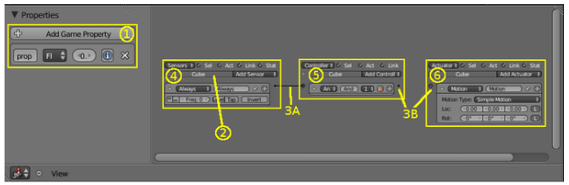
game engine elements



(1) game logic (2) blender game (3) game menu (4) logic editor panel (5) properties

logic editor

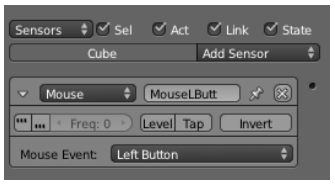
- elements of game logic are defined in Blender as "logic bricks"
- three types of bricks:
  - sensors — listen for events
  - controllers — handle data provided by events
  - actuators — perform actions in response to events



(1) game property area (2) object name (3) links
(4) sensor area (5) controller area (6) actuator area

sensors

- sensors trigger game logic to be activated
- produce output when an event occurs
- sample events: user presses a key or two objects collide
- output is a "pulse" that gets sent to controller(s) connected to the sensor
- a sensor brick looks like this:

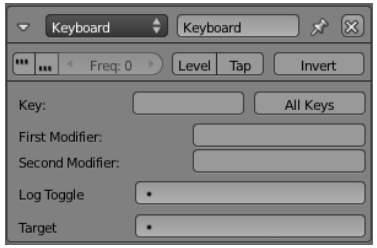


- types of sensors:
  (from `http://wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine/Logic/Sensors`)

Actuator	Detects when a particular actuator receives an activation pulse
Always	Gives a continuous output signal at regular intervals
Collision	Detects collisions between objects or materials
Delay	Delays output by a specified number of logic ticks
Joystick	Detects movement of specified joystick controls
Keyboard	Detects keyboard input
Message	Detects either text messages or property values
Mouse	Detects mouse events
Near	Detects objects that move to within a specific distance of themselves
Property	Detects changes in the properties of its owner object
Radar	Detects objects that move to within a specific distance of themselves, within an angle from an axis
Random	Generates random pulses
Ray	Shoots a ray in the direction of an axis and detects hits
Touch	Detects when the object is in contact with another object

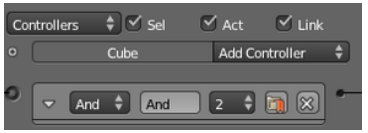
- advanced sensor options:
  - "true" level trigger — causes the sensor to fire "true" pulses, to the attached controller(s)
  - "false" level trigger — causes the sensor to fire "false" pulses, to the attached controller(s)
  - repeating trigger — repeats pulse, based on defined frequency, in units of 60 Hz (every 60th of a second)

- sample keyboard sensor



controllers

- collect data received by sensors
- also can operate based on a specific state
- these are like conditional statements in Java
- a controller brick looks like this:



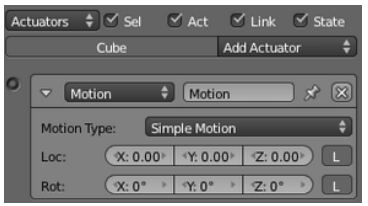
- types of controllers:
  (from `http://wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine/Logic/Controllers`)
  - AND
  - OR
  - XOR
  - NAND
  - NOR
  - XNOR
  - Expression
  - Python

- controller responses based on logic operations on sensor pulse inputs:

number of	*controllers*					
true sensors	AND	OR	XOR	NAND	NOR	XNOR
zero	false	false	false	true	true	true
one (not all)	false	true	true	true	false	false
some (not all)	false	true	false	true	false	true
all	true	true	false	false	false	true

actuators

- actuators perform actions
- an actuator brick looks like this:

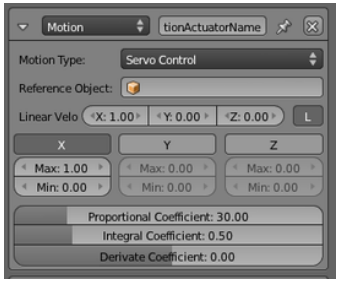


- types of actions:
  (from `http://wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine/Logic/Actuators`)

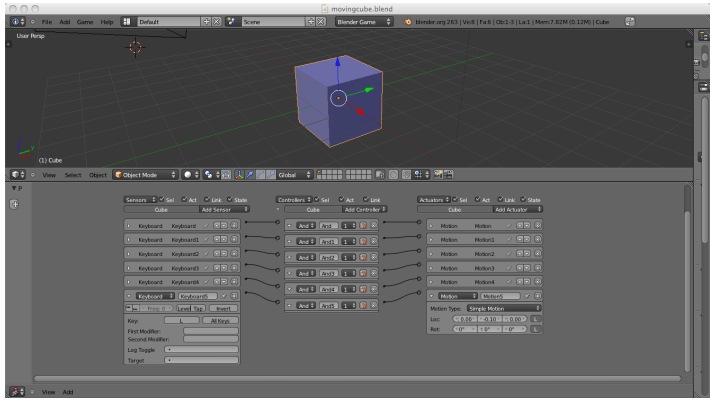
Action	Handles armature actions.
Camera	Has options to follow objects smoothly, primarily for camera objects.
Constraint	Constraints are used to limit objects locations, distance, or rotation.
Edit Object	Edits the objects mesh, adds objects, or destroys them.
Filter 2D	Filters for special effects like sepia colors or blur.
Game	Handles the entire game and can do things as restart, quit, load, and save.
Message	Sends messages, which can be received by other objects to activate them.
Motion	Sets object into motion and/or rotation.
Parent	Can set a parent to the object, or unparent it.
Property	Manipulates the objects properties, like assigning, adding, or copying.
Random	Creates random values which can be stored in properties.
Scene	Manage the scenes in your .blend file. These can be used as levels.
Sound	Used to play sounds in the game.
State	Changes states of the object.
Steering	Provides pathfinding options for the object.
Visibility	Changes visibility of the object.

- sample actuator for simple motion — adjust dx, dy, dz for position and rotation:

- sample actuator for *servo* motion — adjust forces applied to object:



example code: moving cube



example to try

- from `https://sites.google.com/site/blendergameprojects/`,
  by Prof Tim Hickey, Brandeis University (`http://www.cs.brandeis.edu/~tim`)
- Skyracer 2.61
  Designed to let you experiment with the motion actuator for avatars.
  This game kit provides you with a vehicle that can be controlled by *WASD* commands, and you can experiment with the motion actuator panel to give it a better feel. You could also add more jumps, other racetracks, etc.