Game Development: Project Studio **OART-UT 1612 / OART-GT 2612 / NCRD-UT 1612**

2 Metrotech, RM #825. Downtown Brooklyn. TuTh 12:30 – 4:30 PM, July 8 – August 14
INSTRUCTOR: Robert Yang <ry14@nyu.edu> *office hours by appointment*
ASSISTANT: Nolan Filter <nolan.f.filter@gmail.com>

This course reflects the various skills and disciplines that are brought together in modern game development: game design, programming, asset creation, and critical analysis. Classroom lectures and lab time will all be used to bring these different educational vectors together into a coherent whole; the workshop will be organized around a single, long-term, hands-on, game creation project. At the completion of this course, the student will be able to:

1) Describe typical work practice in game development.
2) Demonstrate competency through actual implementation of code and assets.
3) Work with a game engine, and understand the basics of how to build a game in the engine.

YOU WILL NEED:

- A laptop computer! NOT a tablet, NOT an iPhone... but a laptop, PC or Mac.
- Unity, free version. ("Pro" is okay too, but not necessary at all)
- Autodesk Maya (Autodesk offers free 3 year licenses to students)
- Sourcetree (or you can use whichever Git client you prefer, but this is what I'll teach)

WE WILL READ: *(free PDF excerpts will be provided; you may purchase the texts if you wish)*

- 10PRINT, by Nick Montfort, Ian Bogost, et al. The philosophy of code and expression.
- Game Feel, by Steve Swink. The art and science of game input and perception.

LEARNING GOALS: *(the goal is to practice design, code, and asset creation, as a unified discipline)*

- Iterative prototyping processes and troubleshooting, isolating bugs and problems.
- Code literacy, input and control structures (if / else / for / while), basic code patterns.
- Conceptualizing 3D space / raycasting / basic vector math, movement and collisions.
- Basic 3D polygon modeling workflows and considerations, understand 3D as data.

CLASS ASSIGNMENTS:

- WEEKLY HOMEWORK / LABS / READINGS: tasks to train specific techniques / concepts
- MIDTERM “SOLO JAM”: an explorable world with basic logic, individual project
- FINAL PROJECT, “GOLD” RELEASE: a publicly released roguelike game, group project

CLASS WEBSITE: **github.com/radiatoryang/nyu_studio_summer2014/**

To turn-in homework, click "Wiki" on the sidebar, and follow instructions.

ATTENDANCE: 2 unexcused absences will lower your grade, and 3 is grounds for failure.

FINAL PROJECT:
We will work in groups of 4-5 students to build small “roguelike” games.
For the purposes of this class, a roguelike game is: a singleplayer with “permadeath” and procedurally generated terrain / encounters / elements.

- you MUST release your game to the public (for free, or not for free)
- each student MUST complete at least one CODE task, ASSET task, and DESIGN task

SCHEDULE (subject to change)

TUESDAY	THURSDAY	HOMEWORK (due Tues)
7/08: What is game development, student intro, syllabus overview. Unity interface and 3D space. LAB: design thinking, terrain tool	**7/10:** Sandwich challenge, intro to code, Hello World. Building out, Source control with GitHub. LAB: code concepts, SourceTree.	- 10PRINT 10, 15, 20 - poetic landscape, build to web, push project to a Git repo
7/15: Review code concepts, text world. Animal Upon Animal, intro to Maya. Rube Goldberg machine. LAB: code concepts, Maya	**7/17:** Intro to vector math, making character controllers. Start midterm solo jam project. LAB: code concepts, vector math.	- 10PRINT 25, 30, 40 - midterm: make a 3D world with 1 exterior, 1 interior, 1 scripted gate
7/22: Playtest solo jams. Uses and applications for raycasts. Modeling from reference. Particle systems. LAB: playtest, aquarium	**7/24:** Thinking about roguelikes. Procedural generation. Form groups and start final project. LAB: procedural generation	- Game Feel ch. 1 - final: prepare an interface prototype for playtesting
7/29: Playtest prototypes. Various GUI and HUD approaches. Intro to juiciness and game feel tuning. LAB: playtest, feel tuning	**7/31:** Intro to coroutines. Implement juiciness. TA lecture: code architectures. Code review. LAB: shake, arch, code review	- Game Feel ch. 12 - final: prepare a game systems prototype for playtesting
8/5: Playtest prototypes. Alternate 3D modeling workflows and intro to sculpting. LAB: sculpture, playtest	**8/7:** Code review. Working with data and serialization. Making a homemade 3D sculpting tool. LAB: code review, sculpt tool	- source code packets - final: prepare a nearly-done game for last minute playtesting!
8/12: Playtest prototypes. Intro to shaders. Preparing for release. LAB: playtest, glass lab	**8/14:** Final presentations. Pizza day! Intro to virtual reality. LAB: let's barf up pizza to VR	- e-mail and tweet and snapchat about your wonderful new game

ASSESSMENT

Students will be graded primarily on demonstrated process and technique. Students will be given grades based on a 100-point scale. Each assignment will be graded on a point scale, and these points will be added up to determine the final grade, according to the following:

98-100 A+ 92-97 A 90-91 A- 88-89 B+ 82-87 B etc.

The following are the components of the grade:

Attendance & participation 25
Homework 25
Midterm 15
Final 35
TOTAL = 100

Attendance & Participation
The attendance and participation portion of your grade is based on the following:

- Your attendance in class and tardiness. Missing more than 2 classes will hurt your grade.
- Participation in group discussions and critiques
- Peer grades and participation in writing group evaluations

Private peer grades
You'll give a grade to each member of your group. You can add a short explanation if you like, and you must add some explanation when giving a grade of C or below.
A = Fully participated and contributed ideas - hard worker and great teammate
B = Generally was present during the process - no complaints
C = Attended some meetings, but could have contributed more
D = Was absent from most or all meetings, or counter-productive in some way
F = Completely absent from the process

Group evaluations
Students will also write an evaluation of each team member at the end of the class. These evaluations will be sent to all group members and to the instructor. They must include:
a) Two positive observations. Particular skills, behaviors, decisions, or other ways in which a member made a positive contribution.
b) Two areas for improvement. At least two observations that point out how the team member can change their working style, collaborative approach, or other aspects of their behavior to improve the project and the team dynamic.

STATEMENT OF ACADEMIC INTEGRITY
Plagiarism is presenting someone else's work as though it were your own. More specifically, plagiarism is to present as your own: A sequence of words quoted without quotation marks from another writer or a paraphrased passage from another writer's work or facts, ideas or images composed by someone else.
Statement of Principle
The core of the educational experience at the Tisch School of the Arts is the creation of original academic and artistic work by students for the critical review of faculty members. It is therefore of the utmost importance that students at all times provide their instructors with an accurate sense of their current abilities and knowledge in order to receive appropriate constructive criticism and advice. Any attempt to evade that essential, transparent transaction between instructor and student through plagiarism or cheating is educationally self-defeating and a grave violation of Tisch School of the Arts community standards. For all the details on plagiarism, please refer to page 10 of the Tisch School of the Arts, Policies and Procedures Handbook 2013-2014, which can be found online at: http://students.tisch.nyu.edu/page/home.html

ACCESSIBILITY
Academic accommodations are available for students with documented disabilities. Please contact the Moses Center for Students with Disabilities at 212-998-4980 for further information.

Transform

Position, rotation and scale of an object.

Variables

childCount	The number of children the Transform has.
eulerAngles	The rotation as Euler angles in degrees.
forward	The blue axis of the transform in world space.
hasChanged	Has the transform changed since the last time the flag was set to 'false'?
localEulerAngles	The rotation as Euler angles in degrees relative to the parent transform's rotation.
localPosition	Position of the transform relative to the parent transform.
localRotation	The rotation of the transform relative to the parent transform's rotation.
localScale	The scale of the transform relative to the parent.
localToWorldMatrix	Matrix that transforms a point from local space into world space (Read Only).
lossyScale	The global scale of the object (Read Only).
parent	The parent of the transform.
position	The position of the transform in world space.
right	The red axis of the transform in world space.
root	Returns the topmost transform in the hierarchy.
rotation	The rotation of the transform in world space stored as a Quaternion.
up	The green axis of the transform in world space.
worldToLocalMatrix	Matrix that transforms a point from world space into local space (Read Only).

Functions

DetachChildren	Unparents all children.
Find	Finds a child by name and returns it.
GetChild	Returns a transform child by index.
GetSiblingIndex	Gets the sibling index.
InverseTransformDirection	Transforms a direction from world space to local space. The opposite of Transform.TransformDirection.
InverseTransformPoint	Transforms position from world space to local space. The opposite of Transform.TransformPoint.
IsChildOf	Is this transform a child of /parent/?
LookAt	Rotates the transform so the forward vector points at /target/'s current position.
Rotate	Applies a rotation of /eulerAngles.z/ degrees around the z axis, /eulerAngles.x/ degrees around the x axis, and /eulerAngles.y/ degrees around the y axis (in that order).
RotateAround	Rotates the transform about axis passing through point in world coordinates by angle degrees.
SetAsFirstSibling	Move the transform to the start of the local transfrom list.
SetAsLastSibling	Move the transform to the end of the local transfrom list.
SetSiblingIndex	Sets the sibling index.
TransformDirection	Transforms direction from local space to world space.
TransformPoint	Transforms position from local space to world space.
Translate	Moves the transform in the direction and distance of translation.

MonoBehaviour: MonoBehaviour is the base class every script derives from.

CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops all coroutines named methodName running on this behaviour.
print	Logs message to the Unity Console (identical to Debug.Log).

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all game objects when the player gets or loses focus.
OnApplicationPause	Sent to all game objects when the player pauses.
OnApplicationQuit	Sent to all game objects before the application is quit.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	This function is called when the MonoBehaviour will be destroyed.
OnDisable	This function is called when the behaviour becomes disabled () or inactive.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement this OnDrawGizmosSelected if you want to draw gizmos only if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnLevelWasLoaded	This function is called after a new level was loaded.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	OnMouseEnter is called when the mouse entered the GUIElement or Collider.
OnMouseExit	OnMouseExit is called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	OnMouseOver is called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when mouse is released over same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a collider.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the scene.
OnPreCull	OnPreCull is called before a camera culls the scene.
OnPreRender	OnPreRender is called before a camera starts rendering the scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per frame for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called once for each camera if the object is visible.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods is called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Inherited members

Variables

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
animation	The Animation attached to this GameObject (null if there is none attached).
audio	The AudioSource attached to this GameObject (null if there is none attached).
camera	The Camera attached to this GameObject (null if there is none attached).
collider	The Collider attached to this GameObject (null if there is none attached).
collider2D	The Collider2D component attached to the object.
constantForce	The ConstantForce attached to this GameObject (null if there is none attached).
gameObject	The game object this component is attached to. A component is always attached to a game object.
guiText	The GUIText attached to this GameObject (null if there is none attached).
guiTexture	The GUITexture attached to this GameObject (Read Only). (null if there is none attached).
hingeJoint	The HingeJoint attached to this GameObject (null if there is none attached).
light	The Light attached to this GameObject (null if there is none attached).
networkView	The NetworkView attached to this GameObject (Read Only). (null if there is none attached).
particleEmitter	The ParticleEmitter attached to this GameObject (null if there is none attached).
particleSystem	The ParticleSystem attached to this GameObject (null if there is none attached).
renderer	The Renderer attached to this GameObject (null if there is none attached).
rigidbody	The Rigidbody attached to this GameObject (null if there is none attached).
rigidbody2D	The Rigidbody2D that is attached to the Component's GameObject.
tag	The tag of this game object.
transform	The Transform attached to this GameObject (null if there is none attached).
hideFlags	Should the object be hidden, saved with the scene or modifiable by the user?
name	The name of the object.

Functions

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with /tag/?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the game object.

Static Functions

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares if two objects refer to the same.