Survey

Principles and strategies for a maintainable Unity project structure

Description

Dear fellow developers. I'm writing my game design Master thesis about organization and structure strategies for content and code within the Unity editor. In order to have a little best practice overview in the end, I also want to question some developers about their implementations, strategies and views on this topic. The questions are for advanced Unity users and all programming related questions assume you are coding in C-Sharp.

In best case, the survey should be filled out by the tech lead of a project or the responsible person who decides how to organize the workflow in the editor. It should not take more than 15 minutes to fill out all fields. Please, let me know who you are if you don't mind. I could contact you afterwards, if I should have a question. But I also totally understand if you want to stay anonyme.

If you wish, I can send you the final thesis and the anonymized results as a pdf this summer. Just make sure to answer the email field correctly.

Cheers
Goran

My website: http://goga.ch
2017-05-17

Results

Your name or your team / studio name	Individual or team / studio?	How would you describe your position?	What's the Unity version you have worked on your current or last project?	For how long are you working with Unity?	Are you working with version control?	How long will your current project be in development in total (months). If you can't estimate the duration, just guess. Or if you are not working on one, how long did your last project take?
Sand Sailor Studio	Part of a team / studio	Programmer	Unity 5.5.x	1-3 years	Git	Two and a half years
Blindflug Studios AG	Part of a team / studio	Programmer	Unity 5.6.x	3-6 years	Plastic SCM	10
Anonymous 1	Part of a team / studio	Programmer	Unity 5.5.x	1-3 years	Git	4 Months
Anonymous 2	Part of a team / studio	Programmer	Unity 5.6.x	1-3 years	SVN	1 year
Anonymous 3	Part of a team / studio	Tech lead	Unity 5.1.x	more than 6 years	Git	27
Anonymous 4	Part of a team / studio	Tech lead	Unity 5.5.x	1-3 years	SVN	3
Anonymous 5	Part of a team / studio	Technical lead on some projects	Unity 5.5.x	3-6 years	Mercurial and Git	4 months
Anonymous 6	Individual developer	Tech lead	Unity 5.5.x	more than 6 years	Regular Backups	8
Anonymous 7	Individual developer	Programmer	Unity 5.5.x	3-6 years	Git	8 months
Anonymous 8	Individual developer		Unity 5.6.x	1-3 years	Git	
Anonymous 9	Part of a team / studio	Tech lead	Unity 5.5.x	3-6 years	Git	SaaS so it could be another year could be indefinite..
Anonymous 10	Individual developer	Programmer	Unity 5.6.x	1-3 years	Unity collaborate	6
Anonymous 11	Part of a team / studio	Programmer	Unity 5.6.x	3-6 years	Git	18
Anonymous 12	Part of a team / studio	Tech lead	Unity 5.2.x	more than 6 years	Git	30
Anonymous 13	Individual developer	Programmer	Unity 5.5.x	1-3 years	Git in combination with SVN	4 Months
Anonymous 14	Part of a team / studio	Tech lead	Unity 5.6.x	more than 6 years	Git	32
Anonymous 15	Part of a team / studio	Programmer	Unity 5.6.x	3-6 years	Git	15
Anonymous 16	Individual developer	Programmer	Unity 5.6.x	3-6 years	Git	6
Anonymous 17	Individual developer	Game Designer (Mechanics focus)	Unity 5.4.x	1-3 years	No	The one i work now i started a few days back, the last i worked for like a month
Anonymous 18	Individual developer	Tech lead	Unity 5.6.x	1-3 years	SVN	6
Anonymous 19	Individual developer	All of the above + Designer	Unity 5.6.x with continuous integration (updating to new stable version of the engine as soon as it comes out and fixing any project-side issues so I can always access latest features if needed; project started on 4.x)	more than 6 years	Mercurial/Hg	Guesstimate is about 2 more years

Anonymous 20	Individual developer	Programmer	Unity 5.6.x	3-6 years	Collaborate	12
Anonymous 21	Individual developer	Tech lead	Unity 5.4.x	3-6 years	Git	4
Anonymous 22	Individual developer	Programmer	Unity 5.6.x	1-3 years	Git	16
Anonymous 23	Individual developer	Solo developer	Unity 5.6.x	3-6 years	Git	60 months (infrequent development)
Anonymous 24	Part of a team / studio	Tech lead	Unity 5.6.x	3-6 years	Git	36

Your name or your team / studio name	**Could you describe how you name assets and folders in your project view window? (Do you use capital letters or camel case syntax, etc.?)**	**How do you structure your assets in the project view window? How do you separate different assets?**	**How do you name and structure your GameObjects in the hierarchy window inside a scene?**
Sand Sailor Studio	Generally, we do use capitalization on the folder and asset names.	The assets created specifically for the project have their own folder hierarchy within a project root folder placed in the Assets folder. In that hierarchy, each "category" of asset has its own folders: Animations, Models, Scripts, Scenes, etc. Apart from the project root folder, the Assets folder contains additional folders, which usually come from third party unity packages.	We do use the empty GameObject as folders approach, however kept to a minimum, for transform overhead reasons. The folders represent various functional groups within the scene (such as Colliders, Interactable sites, etc). The GameObjects themselves are usually named in a meaningful way for their functional purpose (this applies to prefab instances too), with a capital letter. The only exception from capitalization comes from geometry prefabs (props, etc) which have a lowercase naming mechanism enforced by our prefab synchronization plugin. When the GameObjects contain certain functional components, we do apply specific suffixes to the name (for example the GameObjects which host our visual scripting graphs end with the "Graph" particle, etc).
Blindflug Studios AG	CamelCase grouping with Logic and not File-type. For example: All Classes needed for UI are in the same Folder. All Classes needed for Player are in the same "Scripts/Player" Folder.	Using uniform, standarized GameObjects (localScale = 1,1,1 etc.) as Containers.	Not at all. I don't care about gameObject's names in the Scene. That's totally up to the designers.
Anonymous 1	Folders: 01_FolderName (to put folders in nonalphabetical order. I use both "_" and camelcase for asset naming.	Scripts, 3d Models, Materials, Scenes, Prefabs, Sounds, and Shaders each have a folder on their own.	Objects that are only relevant during a specific stage of the game, typically have the prefix of said stage and are located within an empty object with the name of the stage. Example:StageObjects_Intro>Intro_House. We use that naming convention because our entire game takes place within one unity scene.
Anonymous 2	Always Pascal, no spaces; avoid underscored whenever possible because they are gross, to me.	I have a folder called _Project where everything I write and create goes. That lets me install assets and plugins without moving (and possibly breaking) them. The root of the Assets folder is a clusterfuck, but I just have to got to the top to find my organised folders. My folder is always: _Project ----| Animation ----|----| AnimatorName // Create a folder for each controller and its animations ----|----|----| AnimatorName.controller ----|----|----| AnimationName.anim ----| Materials ----|----| General // Materials that aren't specific to any mesh	_Environment ----- Graphics ----- Lighting Managers Player Camera Canvas

		----| Meshes ----|----| MeshName // Create folder for each mesh to store textures/materials ----| Prefabs ----|----| Entities // If applicable ----|----| Environment ----|----| Items // If applicable ----|----| Managers ----|----| Prototyping ----|----| UI ----| Scenes ----|----| Testing ----| Scripts ----|----| Data // 99% structs 1% actuall functionality ----|----| Editor ----|----| Components ----|----|----| Camera ----|----|----| Debug ----|----|----| General ----|----|----| Movement ----|----|----| Prototyping ----|----|----| UI ----|----| Testing ----| Shaders ----| Sprites ----|----| SpriteGroupName // Make a folder for a collection of sprites ----| Textures ----|----| TextureGroupName // Make a folder for a collection of textures	
Anonymous 3	CamelCase	Depends on the asset, but mostly grouped into logical parts: a model is stored with its textures etc	Some empty objects are used as "folders" to keep things organised. Any runtime-instantiated objects are also placed into folders
Anonymous 4	CapitialisedNames	Plenty of folders, under a single '_GameAssets' folder - keeping separate from system stuff. Use the underscore to keep the folder at the top of the hierarchy. I also use RainbowFolders asset to colorise folders.	CapitializedNames, the structure depends on the scene really, but I try not to go too deep with the hierarchy.
Anonymous 5	We have a naming scheme	Hierarchical, I'd need more time to get into the details, so many things influence this, like Unity specific folder conventions etc.	Hierarchical, IMHO this is quite a broad question
Anonymous 6	Camel Case	Art, Sound, Scripts, Scenes + AssetStore package folders that always go to the root folder (Assets)	Sorted after alphabet, Important stuff starts with an underscore, like "_sceneManager"
Anonymous 7	PascalCase	Audio, models, prefabs, resources, scripts,scenes, sprites, textures	If it's a building built piece by piece, have walls/ceiling/floor gameObjects all be a child of a "building" empty GO. Everything else should be immediately clear what it is by looking at it
Anonymous 8			
Anonymous 9	CamelCase	Seperated into: Scripts, Sprites, Scenes and the broken into relevant categories there	CamelCase

Anonymous 10	Camel case	01-Scene, 02_Script, 03_Prefabs etc.	Depends on the project.
Anonymous 11	CamelCase	First by type e.g Scripts, Sprites, Sounds, and than into categories like Level, Props, Character ect	Main Scene has a Game, Canvas Camera and Level have all parented to one Object
Anonymous 12	Capitalized plain English.	All project-specific assets in _Project. Subfolders organized by asset type.	Functional groupings and geographic groupings.
Anonymous 13	Category/Type as prefix separated by an underline to go on using came case syntax	Kind of depends on the size of the project. Usually create a basic hierarchy of folders, each starting with a number-prefix to keep them sorted: Scenes, Prefabs, Code, Visuals, Controllers, Audio, Fonts. There are usually many subfolders. Differentiate between engine based and project specific code. Using camel case "Prototype" appendix for wip assets.	Scene is usually a mess. Built up a workflow mostly using layers and tags. But constantly using camel case for gameobject names.
Anonymous 14	Mostly camel case, but not strict	Folders for Art, Audio, Scripts,... Subfolders for grouping by topic, setting, usage	Grouping when larger numbers of a specific object exists, grouping by type (trees, trigger, Fx, ...)
Anonymous 15	camel case	they are primarily sorted by type: a folder for scripts, a folder for prefabs, a folder for materials. Within, there are subfolders.	Named in CamelCase, important/more often used objects towards the top of the hierarchy
Anonymous 16	camel case	folders by scope	parent under empty gameobj
Anonymous 17	A mix of suffix and prefix	a pattern for almost all my projects: Animations, Scenes, Prefabs, Textures-Model/Sprites, Materials, Sound, Scripts, and the subdivisions of the previous varies from projetc to project	I use a lot of prefix, to identify every single thing
Anonymous 18	CapitalLetterWithoutSpaceOrUnderScore	Yes, seperate them by type first, then by topic/theme for each type. Like /Texture/Environment	Capital for naming, parent-child for hierachy managing
Anonymous 19	CamelCaseThatStartsWithACapitalLetter (in case of single word folder names I am using small letters only)	All code files written specially for the project and not a part of asset packs goes to Assets/code, textures go into Assets/textures, models into Assets/models, sounds into Asset/sounds, music into Assets/music and so on	I usually name it after what they do, e.g. player character is named Player, any empty object with manager-type script attached to it is named after the script in question and so on. This makes easier to filter them out if needed using filter/search field in the hierarchy window.
Anonymous 20	Camelcase	Third party stuff in one folder and my own stuff in type specific folders	Statics and World geometry in one root object and light ad a sibling, managers in one objekt. Rest loosely
Anonymous 21	CamelCase for files, single word (or spaces) for directories (e.g "Sprites" or "Easy Feedback")	Assets separated into directories by types. (e.g "Scripts," "Sprites," "Textures," etc.) some subfolders where appropriate (e.g. "Scripts/Editor")	CamelCase naming. Parent objects used both for transform parenting, or organization for non-visible game objects
Anonymous 22	Folders are generally all lower case, assets are capitalized camel-case.	scenes at the root. Folders for prefabs, 3dassets, ui, scripts, etc.	Generally I have a GameController and SceneController at the root, and then gameobjects for camera / UI, world, instantiated (bullets, etc) and player.
Anonymous 23	All lower case, snake_case or hyphen-ated, no spaces or special characters.	I have folders under Assets for scripts, images, audio, plugins, editor extensions, and I make subfolders under those if it helps organization.	GameObjects use capitalized CamelCase names. There's usually a Managers object that holds instances of manager scripts, a main camera, a canvas for HUD elements, and then other objects parented as necessary to support transforms as a group. Often all in-scene objects will have a single root empty object as a parent in case they need to be transformed en masse. A lot of my games are UI-only, so all in-scene game objects live under the Canvas object.

Anonymous 24	Folders CamelCase. Prefabs, data, ... most of the time CamelCase. Graphic assets (meshes, images, ...) most of the time lowercase-with-dashes. Some of the assets are prefixed with Types (eg. PrefBullet for a Prefab) or groups (main-menu-icon-star.png) for easier searching. Source files are always CamelCase.	I tend to differentiate and group by context or logical grouping (eg. Prop/Nature/Trees/{tree01.fbx, tree01.png, tree01.mat}) in contrast to type (eg. Materials/..., Textures/...).	CamelCase names. I group objects info logical groups. Parent grouping objects have their transform set to zero if the position/rotation/scale is not important.

Your name or your team / studio name	Do you apply Microsoft's C-Sharp coding conventions in your source?	Do you make use of any dependency injection methods / frameworks?	How do you try to reduce coupling between classes / objects?	How do you cross-reference objects between scenes?	If you work in a team, who is allowed to write own scripts? And is there a review process with a tech lead?	Do you have any programming No-Gos that nobody is allowed to implement?
Sand Sailor Studio	Some of them	We do occasionally use Setters for dependency injections.	Our visual scripting framework is the main provider of events and data chatter between objects (internal graph nodes and normal MonoBehaviour components). There are a number of cases where we do use Unity events.	We do use a combination of singletons, ScriptableObject references and dependency injection (usually applied to the singleton objects), however we do have a very limited amount of objects which need this kind of relation.	The programmers write the C# scripts and the level designers implement some of the functionality on their own using the visual scripting framework. The programmers usually peer-review each and do a small amount of optimization and maintenance work on the visual scripting graphs.	We do try to steer towards a checklist of best programming practices when it comes to C# and Unity and that also doubles as a good measure to keep people from writing blatantly bad code.
Blindflug Studios AG	Some of them	Not really, except for an MVC-approach.	Booth of them	MonoBehaviour singletons	Basically everybody is allowed to write scripts. No standard review process, but for sure one of the techs will have a look at it and decide whether it's useful or not.	GetComponent() is not allowed to in Update()-like methods.
Anonymous 1	Some of them	I use Interface, Setter and Constructor Injector method. I try to limit myself to one per project. So if I start using Setter Injection, I'll constantly use it through the entire project where possible.	Coming from oldschool Java, I typically setup a System myself. Defenitely something I need to look into.	MonoBehaviour singletons	I only work in small teams but Typically I review the code with the respective team members to assure consinstency.	Generally everything that is contradictionary to the general paradigm of what i setup. Not a huge fan of coroutines as unexperienced programmers tend to implement logic mistakes on how stuff is sequenced.

Anonymous 2	Some of them	I use basic dependency injection with interfaces	Unity events	MonoBehaviour singletons	There are two of use. whenever someone writes a script, the other person reviews it with them.	1. We pretty much never have more than one or two update methods 2. Lerp (transform.position, target.position, Time.deltaTime); 3. public void RefreshListOfSomething (List<ListData> baseOnThis) { DestroyCurrentList (); RecreateListBasedOnNewData (); }
Anonymous 3	Yes		Interfaces, C# events, SRP, whatever's appropritate	MonoBehaviour singletons	Yes, absolutely everything goes through pull requests and code review	SendMessage
Anonymous 4	Some of them	Nope	I dont really have the need to do so.	I keep references in ScritableObjects.		
Anonymous 5	Yes, and we use resharper to enforce them	We have our own system framework and system provider which manages dependencies	System framework	Single scene and everything is loaded dynamically, scenes are a pain	Every developer, yes we have code reviews though no formalization yet	We have coding guidelines, everybody adheres to those
Anonymous 6	No	no	no	non monobehaviour singletons	-	-
Anonymous 7	Some of them	No	Unity events	MonoBehaviour singletons		
Anonymous 8						
Anonymous 9	Yes	Only one we maintain	Unity events	MonoBehaviour singletons	Anyone can write a script, all reviewed by tech lead	"hack" methods, if its being thrown together quickly it gets rejected
Anonymous 10	Some of them	Yes, not using a framework though.	No events, delegates, observer pattern	I never have to.	Everybody. There is no review, but i really would prefere working with weekly reviews.	Big classes, animation events, messages, basicly everthing which is hard to debug, where its hard to find out from where something is executed.
Anonymous 11	I don´t know	No	separation of data and logic (ECS)	I never have to.	The people who are able. No	A few syntactical rules but otherwise no

Anonymous 12	Some of them	No.	Booth of them	Static wrappers and ScriptableObjects.	Anyone. Tech lead reviews.	UnityScript.
Anonymous 13	Yes	Yes? Why would anyone not? Always glad using packaged dependencies of huge frameworks. Cleanest way for third party stuff too.	Using .NET C# events and by building a clever modular state engine. Reflection is your friend ;) .	I don't really like Unitys scene handling. I generally use MonoBehaviours singletons or Unitys ScriptableObject. But especially when it comes to a bigger project, I tend to avoid cross-references and build up a load-into-scene level system.	Everybody. Source control -> everyone gets at least one own branch. Review process of pull requests and scrum meetings face-to-face.	Do not touch that freaking master branch! Oh... yes, and do not implement flash, please. It's also very project specific. Networking is usually a thing that should be monitored in detail before implementation.
Anonymous 14	Some of them	No, to complicated. However using these patterns to some extent	custom event system	MonoBehaviour singletons	Everybody can write scripts, all reviewed by me ;) rarely happens	Yes, public variables, unnecessary inspector references,
Anonymous 15	Some of them			I keep references in ScritableObjects.		
Anonymous 16	Yes	no	Booth of them	MonoBehaviour singletons	not in a team	no
Anonymous 17	Some of them		Booth of them	I keep references in ScritableObjects.		
Anonymous 18	Some of them	No	Booth of them	MonoBehaviour singletons		
Anonymous 19	Some of them		Unity events	Static wrappers / accessor for MonoBehaviour scripts		
Anonymous 20	No	No	Unity events	MonoBehaviour singletons		Non optimized runtime code. And editor code without compileflags
Anonymous 21	Some of them	Yes	Booth of them	MonoBehaviour singletons		
Anonymous 22	Some of them	No	Unity events	I have a GameController gameobject that is maintained across scenes		
Anonymous 23	No	No	Booth of them	I never have to.	Solo developer	Not really.
Anonymous 24	Some of them	Yes, but not excessively. No framework, own code.	Own message bus, pub/sub system.	All of the above depending on the context.	People are allowed to write own scripts and should respect the already existing way of doing things. The less tech savvy people are the more review and feedback is necessary. It also depends on how critical the area is. We do more or less regular profiling sessions of the game.	No, but the general rule that one should write code that fits the existing one and not reinvent everything every time.

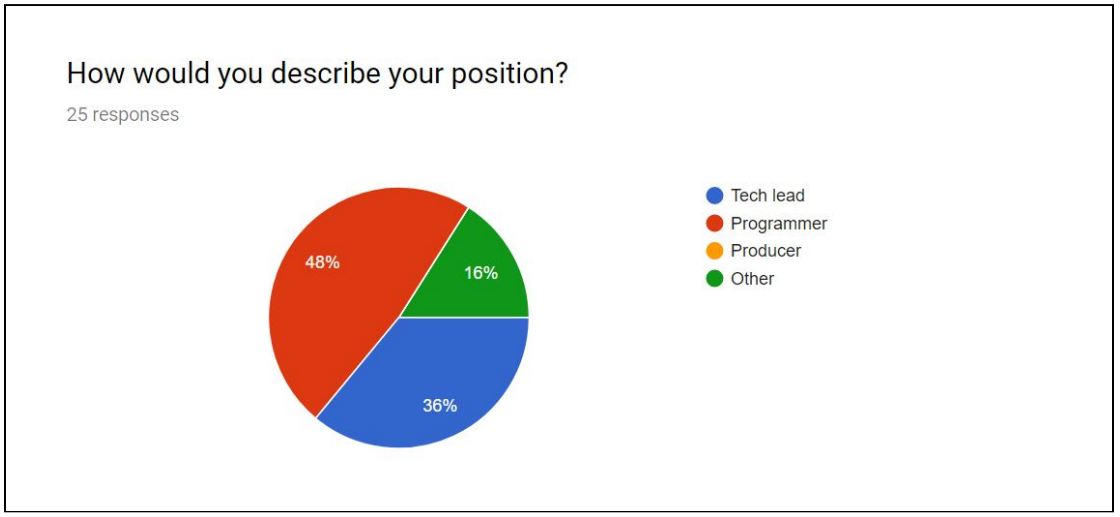
Your name or your team / studio name	Are you working with the multiple scene editing feature?	If not, do you prevent Unity to load all your assets at the initialization of your game?	How do you make use of multiple scene editing in Unity?	Are you using any third-party tools to achieve extended functionality?	How do you cluster your scenes?	Are you loading scenes asynchronously?
Sand Sailor Studio	Yes		The level designers sometimes need adjacent scenes loaded together, in order to properly setup the camera transitions which happen when crossing scenes.	Not for multi-scene editing.	Playable levels are split into gameplay scenes and static background geometry. Each playable level will be loaded additively, according to player position.	Yes
Blindflug Studios AG	No	No, because Unity will only load the assets needed for/referenced in the first/initial scene. There's no need to prevent Unity from loading everything else.				
Anonymous 1	No	I work with pooling where possible to reduce overhead. But in the current project, everything is loaded at the start of the game and then deactivated to reduce performance loss.				
Anonymous 2	No	No, wasn't aware of that option				
Anonymous 3	No	Yes				
Anonymous 4	Yes	Not really, N/A for my games in general.	a 'Setup' scene which every other scene loads in, so that the game can be initialized from any scene.	AdvancedInspector, uTomate.	Scenes represent new content / levels that will be loaded additionally according to the position of the player	Some
Anonymous 5	No	No				
Anonymous 6	Yes	-	To speed up startup times I just load the main menu in a small scene and the game additionally afterwards while the user clicks around the main menu.	Lidgren networking and some mono DLLs	One Level = One Scene	Yes

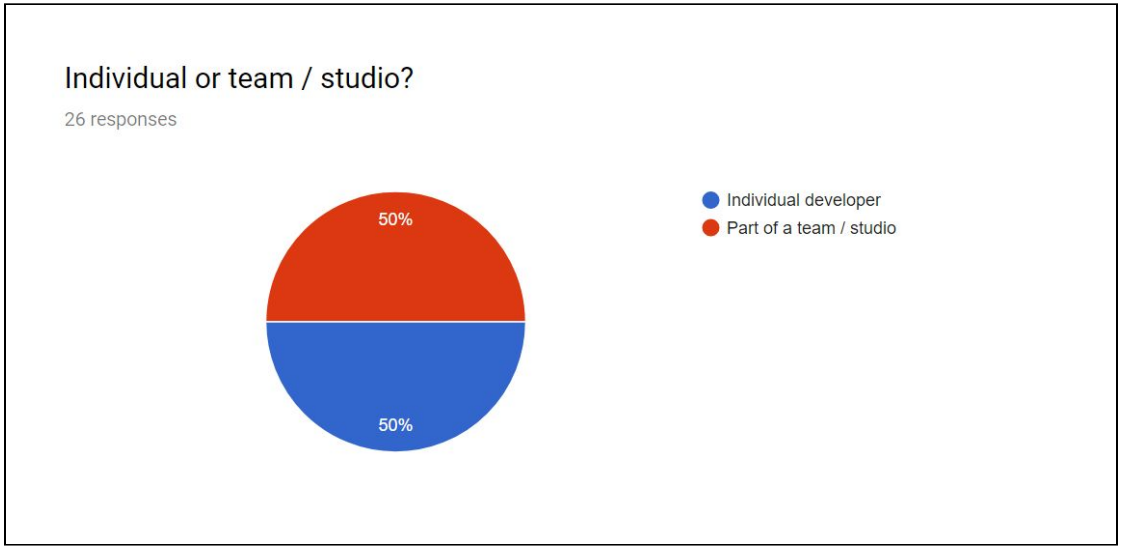
Anonymous 7	No	No				
Anonymous 8						
Anonymous 9	Yes		Often used in order to quickly change large areas of the project, functionally we use multiple scenes + scene merging in order to add new parts of the game at run time	no	Separate logic from content	Yes
Anonymous 10	No	Worldtrigger, assetbundles, async asset loading etc.				
Anonymous 11	No	Yes by aditive loading additional scenes				
Anonymous 12	No	Yes.				
Anonymous 13	No	I didn't need to do so til now.				
Anonymous 14	Yes		Load connected / adjacent scenes as reference (disabled editing)	Not for multi scene editing	Scenes represent new content / levels that will be loaded additionally according to the position of the player	Yes
Anonymous 15	No	not yet				
Anonymous 16	Yes			yes	Separate scenes, so that multiple disciplines can work together on a scene (audio, visuals, coding, etc.)	Yes
Anonymous 17	No	No				
Anonymous 18	No	Yes				
Anonymous 19	No					
Anonymous 20	No	No				
Anonymous 21	Yes		Load "required objects" scene with scripts/objects/data for the scene being worked on/tested	No	Scenes represent new content / levels that will be loaded additionally according to the position of the player	Yes
Anonymous 22	No	No				
Anonymous 23	No	No.				
Anonymous 24	Yes		Bootstrap, Core scenes that add additional scenes (eg. level, ui, parallax backgrounds) to be able to edit them separately and easy unloading/cleaning of multiple objects and assets.	No.	Separate logic from content, Separate scenes, so that multiple disciplines can work together on a scene (audio, visuals, coding, etc.), Scenes represent new content / levels that will be loaded additionally according to the position of the player, resource management, easy unloading of groups of stuff	Yes

Your name or your team / studio name	Do you need Unity's scene editor to place content, functionality, lights etc?	Do your level designers need to script code? Options: 1.Yes, they can write their own scripts for a specific scenery. 2. No, they have visual tools or components to connect functionality. 3. Other	How do you animate things in your level? Options: 1. Imported keyframe / bone animations from 3d tools 2. With Unity's keyframe animation tools 3. Third-party tools like iTween etc., or own written tools 4. Other
Sand Sailor Studio	Yes	No, they have visual tools or components to connect functionality.	A mix between these options
Blindflug Studios AG	Depends on the project. There's no solution that fits every project-type.	No, they have visual tools or components to connect functionality.	A mix between these options
Anonymous 1	Yes	both	Imported keyframe / bone animations from 3d tools
Anonymous 2	Yes	No, they have visual tools or components to connect functionality.	Custom tweening and sometimes Unity animation for more complex UI stuff.
Anonymous 3	Yes	No, they have visual tools or components to connect functionality.	A mix between these options
Anonymous 4	Yes, but I try to avoid to depend too much on the scene editor.	Sometimes, but they're experienced.	A mix between these options
Anonymous 5	Yes, but I try to avoid to depend too much on the scene editor.	No level design needed specifically, depends on project	A mix between these options
Anonymous 6	Yes	Yes, they can write their own scripts for a specific scenery.	Imported keyframe / bone animations from 3d tools
Anonymous 7	Yes	No, they have visual tools or components to connect functionality.	Imported keyframe / bone animations from 3d tools
Anonymous 8			
Anonymous 9	Yes, but I try to avoid to depend too much on the scene editor.	No, they have visual tools or components to connect functionality.	With Unity's keyframe animation tools
Anonymous 10	Yes, but I try to avoid to depend too much on the scene editor.	Both	Imported keyframe / bone animations from 3d tools
Anonymous 11	Yes	Generated or i´m the leveldesigner	Spine and Dotween
Anonymous 12	Yes	No, they have visual tools or components to connect functionality.	A mix between these options
Anonymous 13	Yes	Both ;)	Imported keyframe / bone animations from 3d tools
Anonymous 14	Yes	No, they have visual tools or components to connect functionality.	A mix between these options
Anonymous 15	Yes, but I try to avoid to depend too much on the scene editor.	sometimes, but rarely	A mix between these options
Anonymous 16	Yes	Yes, they can write their own scripts for a specific scenery.	A mix between these options
Anonymous 17	Yes	I work alone, so if its need, i do script, but i try to avoid put a lot of scene especific scripts, and find a way to make a single script to all needs of the scene, and put in a controller	A mix between these options

Anonymous 18	Yes	No, they have visual tools or components to connect functionality.	Imported keyframe / bone animations from 3d tools
Anonymous 19	No, we wrote our own level editor.	No, they have visual tools or components to connect functionality.	A mix between these options
Anonymous 20	Sometimes	Im just me...	A mix between these options
Anonymous 21	Yes	Yes, they can write their own scripts for a specific scenery.	With Unity's keyframe animation tools
Anonymous 22	Yes, but I try to avoid to depend too much on the scene editor.	No, they have visual tools or components to connect functionality.	With Unity's keyframe animation tools
Anonymous 23	Yes	Yes, they can write their own scripts for a specific scenery.	Third-party tools like iTween etc., or own written tools
Anonymous 24	We use the unity editor as some kind of level editor but export the level into our own format and load it from another "empty" scene. Multiplayer game where the server does not depend on Unity.	No, they have visual tools or components to connect functionality.	Imported keyframe / bone animations from 3d tools

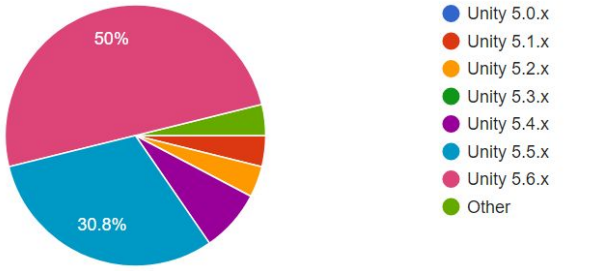
Charts


How would you describe your position?
25 responses
48%
16%
36%
Tech lead
Programmer
Producer
Other


Individual or team / studio?
26 responses
50%
50%
Individual developer
Part of a team / studio

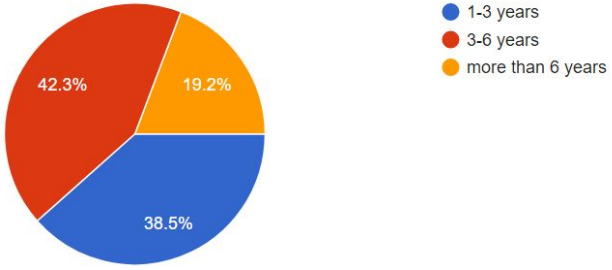
What's the Unity version you have worked on your current or last project?

26 responses



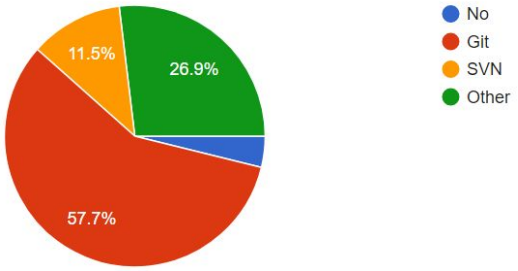
For how long are you working with Unity?

26 responses



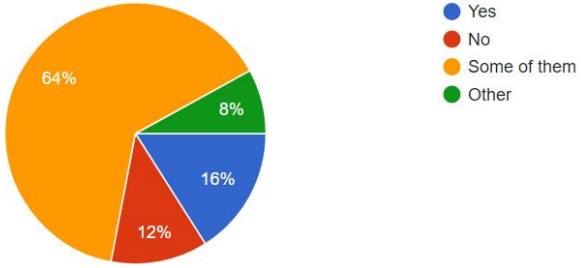
Are you working with version control?

26 responses



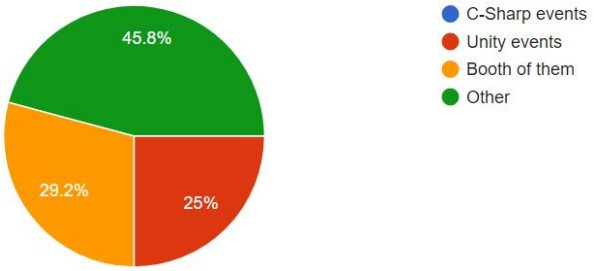
Do you apply Microsoft's C-Sharp coding conventions in your source?

25 responses



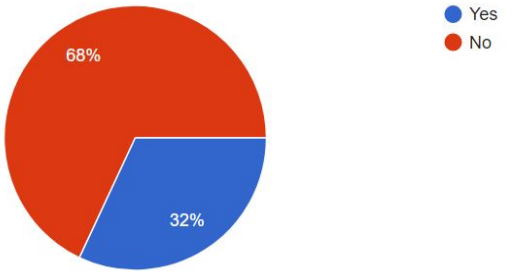
How do you try to reduce coupling between classes / objects?

24 responses



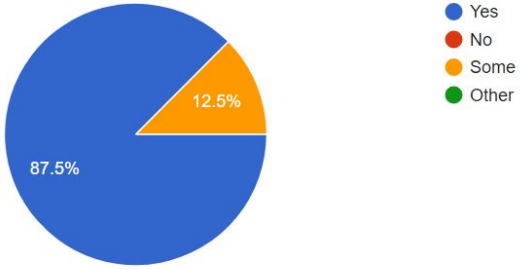
Are you working with the multiple scene editing feature?

25 responses



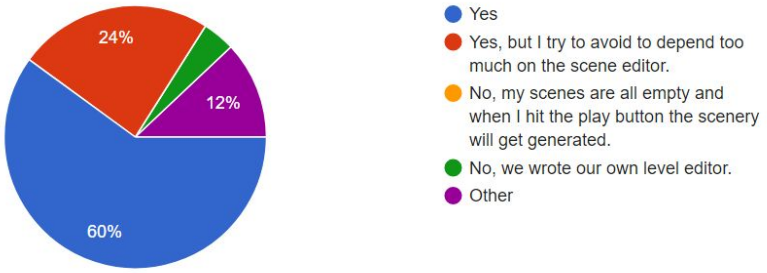
Are you loading scenes asynchronously?

8 responses



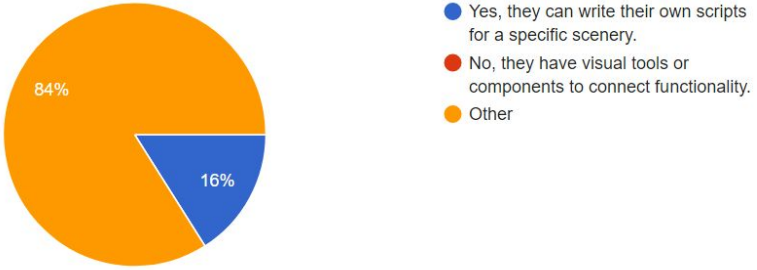
Do you need Unity's scene editor to place content, functionality, lights etc?

25 responses



Do your level designers need to script code?

25 responses



How do you animate things in your level?

25 responses