C# for Java Developers Crash Course

PAL Val | Discord: @valknight.xyz PAL Jude | Discord: @jude_birch

who is this for?

who is this for?

Fairly new to C#

Familiar with Java*

* or any other object-oriented & strongly typed language

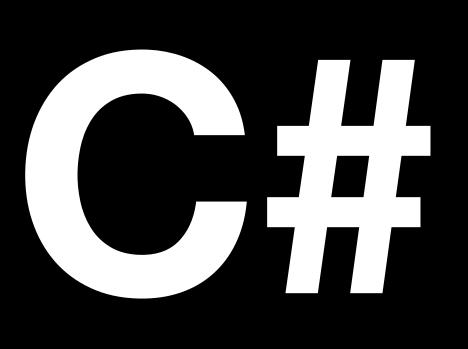
Wanting to make games using C#

* see Unity, Godot, MonoGame, FNA, etc etc etc

on the Java note

a quick warning in advance





MICROSOFT'S JAVA

things you can keep if you already know java

Java Virtual Machine -> Common Language Runtime Most of the syntax Lots of the same OO language features Garbage collection (we'll be talking about this!)

>> this session will assume you are semi-comfortable with Java principles!



on the games note

last warning, then content, i swear



this is written with Unity in mind

this is written with Unity in mind

but nearly all of this is transferable

this is written with Unity in mind

but nearly all of this is transferable and for gamedev folks, you'll be using Unity as part of your degree

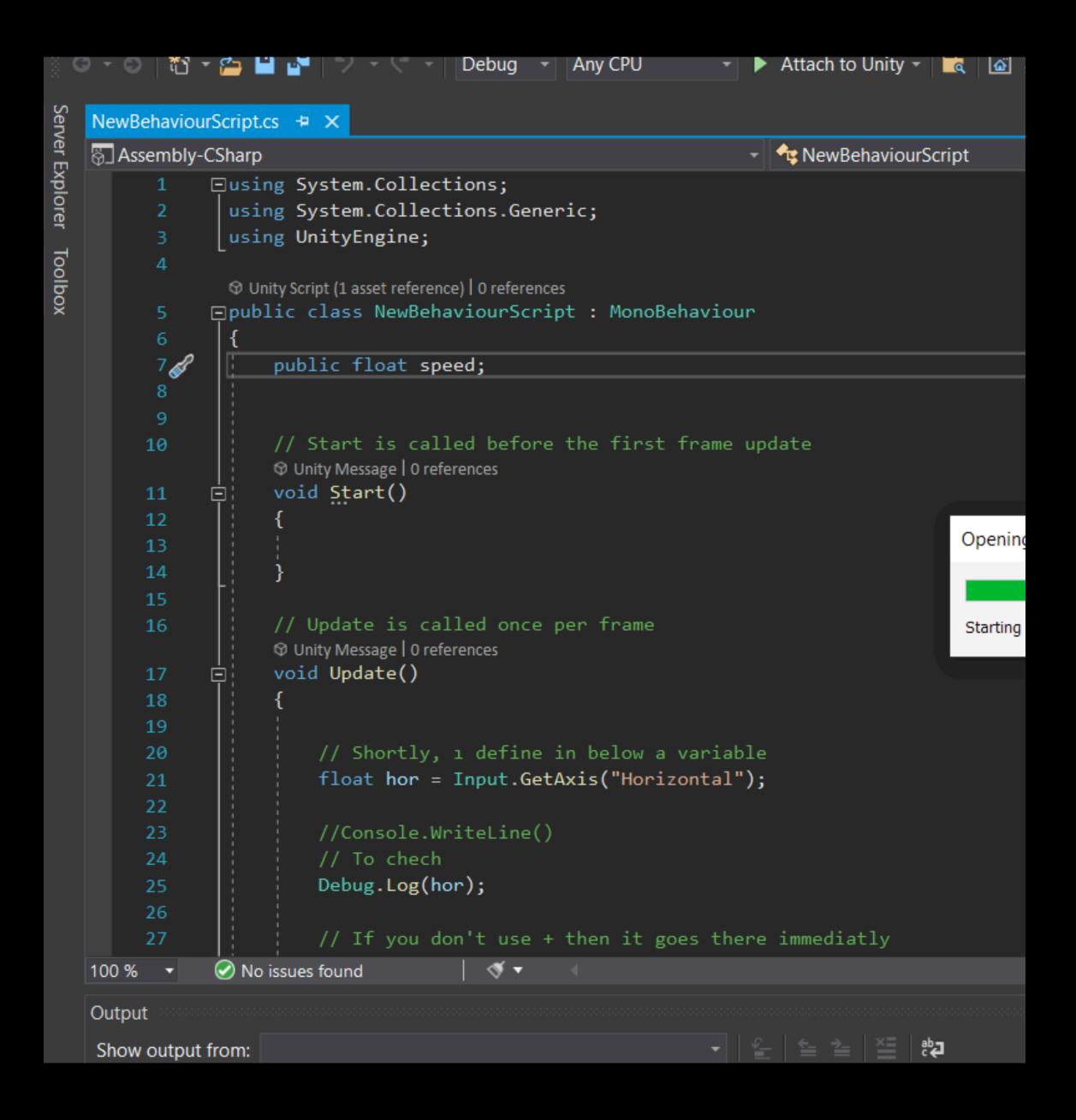
C# Fundamental 0: What IDE do I use?

C# Fundamental 0: Which of the three different Visual Studios should I use?

Visual Studio Windows

- + Free below a revenue cap
- + The OG C# IDE
- + Great Unity integration OOB
- + Lots, and lots of features

- Windows only
- Quite heavy
- Lots, and lots of features



Visual Studio for Mac macOS

- + Free below a revenue cap
- + Good Unity integration OOB
- + Quick to get started with
- Already deprecated by Microsoft
- Buggy and clunky

- Lacking in a lot of features compared to it's Windows counterpart

- Not really Visual Studio - it's just an old IDE called Xamarin Studio, which itself was an older IDE called MonoDevelop

● ● ● ► □ Podcast.Server > D	ebug → [] Microsoft Edge		Podcast.Web
Solution	\star 🗸 < > 🔹 FloatingPlayer.razor	CategoriesPage.raz	or.css CategoriesPage.razo
 Podcast.Web Podcast.Client Podcast.Components Connected Services Dependencies - 2 updates Enums 	174 : 1 175 176 private 177 OnF 178 179 private	PlaybackRateChange.Inv	hanged(double value) => okeAsync(playbackRate = va ume(ChangeEventArgs e)
> Events			
> Extensions	Solution		K FloatingPlayer.razor Cat
> 🛅 ListenTogether	✓ ■ Podcast.Web	173	-
> 🛅 Models		174	
✓ Image Player	> Podcast.Client	170	
> FloatingPlayer.razor	Podcast.Components	17	
> Shared	Connected Services	178	
	> Dependencies – 2 updates	179	
> in www.root	> 🛅 Enums	180	
Imports.razor	> Events	182	
AudioInterop.cs	> Extensions	183	
ClipboardInterop.cs		184	await On
IAudioInterop.cs	> ListenTogether	185	
ListenTogetherHubClient.cs	> 🛅 Models	180	
1 LocalStorageInterop.cs	🗸 📄 Player	18	
	> EloatingPlayer.razor	189	
() ThemeInterop.cs	> 📄 Shared	190	
V Podcast.Pages	> www.root	193	
Connected Services		192	
> 🛅 Dependencies – 1 update	C _Imports.razor	193	
🗸 🛅 Data	() AudioInterop.cs	195	
() ListenLaterService.cs	ClipboardInterop.cs	190	1
PlayerService.cs	IAudioInterop.cs	197	
	ListenTogetherHubClient.cs	198	
O SubscriptionsService.cs	D LocalStorageInterop.cs	199	
V Events	0 ThemeInterop.cs	200	
DurationChangeEventArgs.cs	V Podcast.Pages	202	3
EventHandlers.cs	_	203	
PlaybackRateChangeEventArgs.cs	Connected Services	204	
	> Dependencies - 1 update	205	
	🗸 🚞 Data	20	
	 ListenLaterService.cs 	208	
	PlayerService.cs	209	
	[1] SubscriptionsService.cs	210	
	✓ ■ Events	211	
		213	
	① DurationChangeEventArgs	s.cs	· ·

O PlaybackRateChangeEventArgs.cs

PlayerService. alue); Podca tegoriesPage.razor.css kRateChanged(playba pletedTask; nPlaybackRateChanged RateChange.InvokeAsy ask ChangeVolume(Ch != null && int.Try = volume; nVolumeChange.Invok ume == 0 && !IsMute ait OnMutedChange.In (Volume >= 0 && Is ait OnMutedChange.In nInputCurrentTime(Ch != null && int.Try

!= null && int.Try SetCurrentTime(time) = null;

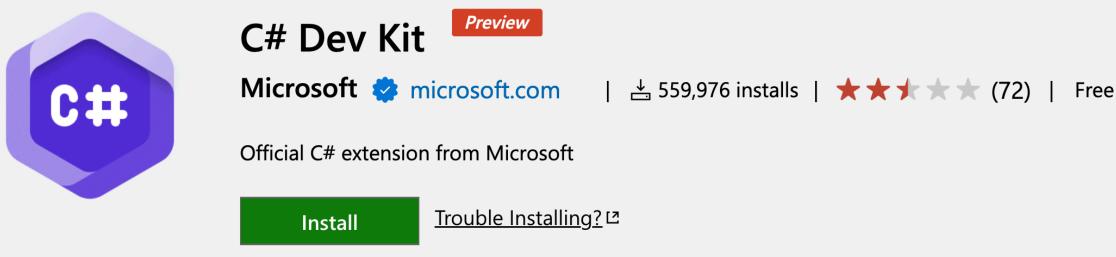
aTime = time: ask OnChangeCurrent

Visual Studio Code Windows, macOS, Linux

- + Base editor is free and open source
- + The only supported option on Linux (and soon to be macOS...)
- + Extension support can keep your one IDE & workflows for C#, Python, JS...
- Unity support is still in preview, and is still a little rough around the edges
- Requires installation of extensions for C#

- Extension is *not* open source, still subject to same VS license, for less features than the other VS IDEs...

File	Edit Selection View Go Run …	←	→	
	UN AND DEBUG D.NET Core Launch (con: ~	\$ <u>.</u>	େ HelperTests.cs M × 🛛 🗄 🕩 😤 🏌 🗘 ୪୦୦ 🖉 ×	
	ARIABLES		MathTests > 😋 HelperTests.cs > 😫 HelperTests > 🗘 TestWriteMessage	
	Locals Numer (textsed) (100%)		18	
	<pre>this: {MathTests.HelperTests}</pre>		20 [TestMethod]	
ı ,	<pre>msg [string]: "This is a test!"</pre>		0 references	
	<pre>writer [StringWriter]: {This is a tes</pre>	t!}	<pre>_21public void TestWriteMessage() {</pre>	
>	reader: {System.IO.StringReader}) 22 m Tet String msg = "This is a test!";	
	<pre>consoleOutput [string]: null</pre>		<pre>23 using StringWriter writer = new();</pre>	
			<pre>24 using StringReader reader = new(msg); 25 Console.SetOut(writer);</pre>	
			26 Console.SetUn(reader);	
			27 Helpers.WriteMessage(msg);	
		tem.int32[<pre>28 Http://www.seconsoleOutput = writer.ToString();</pre>	
			<pre>29 Assert.AreEqual(msg, consoleOutput.TrimEnd());</pre>	
			30 }	
\sim w	ATCHD TestAddPerformOperation		31 Jan v meda der Link	
			32 [TestMethod]	
		n.Int32III	0 references	
		n Int32III	<pre>> 33 public void TestGetOptionInput() { 34 String msg = "4";</pre>	
			34 String msg = 4; 35 using StringReader reader = new(msg);	
			36 Console.SetIn(reader);	
			<pre>37 int result = Helpers.GetOption("Select the operation to perform: ");</pre>	
$\sim c$	V CALL STACK		<pre>38 Assert.AreEqual(int.Parse(msg), result);</pre>	
>	.NET ThreadPool Worker	PAUSED	39	
>	Main Thread	PAUSED	40 }	
>	.NET ThreadPool IO	PAUSED	41	
>	.NET Timer	PAUSED		
>	.NET Long Running Task	PAUSED	PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS	





Visual Studio Windows

Visual Studio for Mac macOS

Visual Studio Code Windows, macOS, Linux

Visual Studio Windows

Visual Studio for Mac *macOS*

Visual Studio Code Windows, macOS, Linux



A New fee has appeared

CHALLENGER APPROACHING



+ Free while you're a student, or if you use the Nightly build

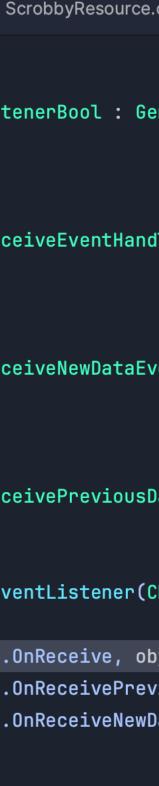
	lacksim		
	Solution \checkmark	dModulo.cs	c# ScrobbyDependency.cs C# S
4	 Sprout · 3 projects C# Scrobby 	12	⊠ 13 usages 🛛 & Val Knight
00	 > & Dependencies > Generated > C Scrobby 	13 îo 14	<pre>public partial class EventListe {</pre>
	 > EventInvoking > EventInvokingData 	15	[Signal] & Val Knight
	 EventListening C# EventListenerBool.cs C# EventListenerDouble.cs C# EventListenerFloat.cs C# EventListenerGodotCollectionsArray.cs C# EventListenerGodotCollectionsArraybo 	19 20 21	public delegate void OnRece [Signal] & Val Knight public delegate void OnRece [Signal]
	C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerCodotCollectionsArrayGo Bookmarks	22 23	La Knight public delegate void OnRece I+4 usages
		24 ^ 25	protected override void Eve {
	Sprout ■ Pinned in Debugger	25 26 % 27 28 29	<pre> EmitSignal(SignalName.C EmitSignal(SignalName.C EmitSignal(SignalName.C } </pre>
		ሻወ	1 F

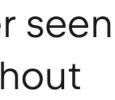


Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante, Unity CTO & Founder

പ്പ





+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

	lacksim		
	Solution \checkmark	dModulo.cs	c# ScrobbyDependency.cs C# S
-4-	 Sprout · 3 projects C# Scrobby 	12	🗷 13 usages 🛛 🚨 Val Knight
00	 > & Dependencies > Dependencies > Generated > Scrobby 	13 îo 14	<pre>public partial class EventList {</pre>
	 > D EventInvoking > D EventInvokingData 	15	[Signal] & Val Knight
•••	 EventListening C# EventListenerBool.cs C# EventListenerDouble.cs 	16 17	public delegate void OnRece
	C# EventListenerFloat.cs C# EventListenerGodotCollectionsArray.cs C# EventListenerGodotCollectionsArraybc		[Signal] & Val Knight public delegate void OnRece
	c# EventListenerGodotCollectionsArraydo c# EventListenerGodotCollectionsArraydo	20	[Signal]
	c# EventListenerGodotCollectionsArrayGo c# EventListenerGodotCollectionsArrayGo c# EventListenerGodotCollectionsArrayGo	22	2 Val Knight public delegate void OnRece
	C# EventListenerGodotCollectionsArrayGo	23	⊡ 1+4 usages 🛛 🚨 Val Knight
	Bookmarks	24 ^o 25	protected override void Eve {
	■ Sprout ■ Pinned in Debugger	26 %	EmitSignal(SignalName.(
		27 28	EmitSignal(SignalName.(EmitSignal(SignalName.(
		29 30	}

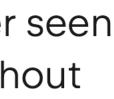


Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante, Unity CTO & Founder

പ്പ





+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

+ Supports cutting edge Unity features like ECS, Burst

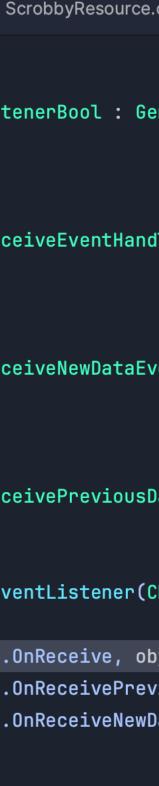
	\bullet \leftarrow \rightarrow S Sprout \checkmark $^{\circ}$ master \checkmark		
	Solution \sim	dModulo.cs	c# ScrobbyDependency.cs C# S
-0-	 Sprout · 3 projects C# Scrobby 	12	⊡ 13 usages 🛛 & Val Knight
00	 > & Dependencies > Dependencies > Dependencies > Contracted > Contracted 	13 îo 14	<pre>public partial class EventListe {</pre>
	 > D EventInvoking > D EventInvokingData 	15	[Signal] & Val Knight
•••	 EventListening C# EventListenerBool.cs C# EventListenerDouble.cs C# EventListenerFloat.cs C# EventListenerGodotCollectionsArray.cs C# EventListenerGodotCollectionsArraybo 	19	public delegate void OnRece [Signal] & Val Knight public delegate void OnRece
	C# EventListenerGodotCollectionsArraydo C# EventListenerGodotCollectionsArrayfo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo	21 22 23	[Signal] ≗ ValKnight public delegate void OnRece
	Bookmarks	24 ^o 25	I+4 usages ≗ Val Knight protected override void Even {
	■ Sprout ■ Pinned in Debugger	26 % 27 28 29	EmitSignal(SignalName.(EmitSignal(SignalName.(EmitSignal(SignalName.(}
		30	3

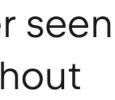


Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante, Unity CTO & Founder

ഴ





+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

+ Supports cutting edge Unity features like ECS, Burst

+ Shader support + Performance warnings as part of IntelliSense warnings

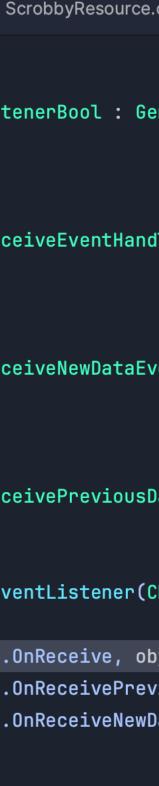
	lacksim		
	Solution \checkmark	dModulo.cs	c# ScrobbyDependency.cs C# S
- 6-	 Sprout · 3 projects C# Scrobby & Dependencies 	12 13 îo	⊡ 13 usages ≗ Val Knight public partial class EventListe
	 Generated Scrobby 	14	{
□ .:	 > D EventInvoking > D EventInvokingData > D EventListening 	15 16	[Signal] & Val Knight public delegate void OnRece
	C# EventListenerBool.cs C# EventListenerDouble.cs C# EventListenerFloat.cs C# EventListenerGodotCollectionsArray.cs	17 18	[Signal] & Val Knight
	C# EventListenerGodotCollectionsArrayboC# EventListenerGodotCollectionsArraydoC# EventListenerGodotCollectionsArrayflo	20	public delegate void OnRece [Signal]
	C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo	22	& Val Knight public delegate void OnRece
	Bookmarks		⊡ 1+4 usages ≗ Val Knight protected override void Eve
	≣ Sprout	25	-{
	■ Pinned in Debugger	26 X	EmitSignal(SignalName.(
		27	EmitSignal(SignalName.(
		28	EmitSignal(SignalName.(
		29 30	} 2
		211	

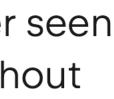


Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante, Unity CTO & Founder

ഴ





+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

+ Supports cutting edge Unity features like ECS, Burst

+ Shader support + Performance warnings as part of IntelliSense warnings

+ Integration with the Unity Editor and **Unity Documentation**

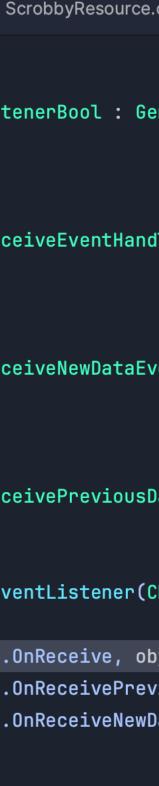
	\bullet \leftarrow \rightarrow S Sprout \checkmark $^{\circ}$ master \checkmark		
	Solution \sim	dModulo.cs	c# ScrobbyDependency.cs C# S
-0-	 Sprout · 3 projects C# Scrobby 	12	⊡ 13 usages 🛛 & Val Knight
00	 > & Dependencies > Dependencies > Dependencies > Contracted > Contracted 	13 îo 14	<pre>public partial class EventListe {</pre>
	 > D EventInvoking > D EventInvokingData 	15	[Signal] & Val Knight
•••	 EventListening C# EventListenerBool.cs C# EventListenerDouble.cs C# EventListenerFloat.cs C# EventListenerGodotCollectionsArray.cs C# EventListenerGodotCollectionsArraybo 	19	public delegate void OnRece [Signal] & Val Knight public delegate void OnRece
	C# EventListenerGodotCollectionsArraydo C# EventListenerGodotCollectionsArrayfo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo	21 22 23	[Signal] ≗ ValKnight public delegate void OnRece
	Bookmarks	24 ^o 25	I+4 usages ≗ Val Knight protected override void Even {
	■ Sprout ■ Pinned in Debugger	26 % 27 28 29	EmitSignal(SignalName.(EmitSignal(SignalName.(EmitSignal(SignalName.(}
		30	3

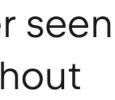


Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante, Unity CTO & Founder

ഋ





+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

+ Supports cutting edge Unity features like ECS, Burst

+ Shader support + Performance warnings as part of IntelliSense warnings

+ Integration with the Unity Editor and Unity Documentation

+ Recommended by one of the cofounders of Unity

	\bullet \leftarrow \rightarrow S Sprout \checkmark $^{\circ}$ master \checkmark		
	Solution \checkmark	dModulo.cs	c# ScrobbyDependency.cs C# S
4	 Sprout · 3 projects C# Scrobby 	12	⊠ 13 usages 🛛 & Val Knight
00	 > & Dependencies > Generated > C Scrobby 	13 îo 14	<pre>public partial class EventListe {</pre>
	 > EventInvoking > EventInvokingData 	15	[Signal] & Val Knight
	 EventListening C# EventListenerBool.cs C# EventListenerDouble.cs C# EventListenerFloat.cs C# EventListenerGodotCollectionsArray.cs C# EventListenerGodotCollectionsArraybo 	19 20 21	public delegate void OnRece [Signal] & Val Knight public delegate void OnRece [Signal]
	C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerCodotCollectionsArrayGo Bookmarks	22 23	La Knight public delegate void OnRece I+4 usages
		24 ^ 25	protected override void Eve {
	Sprout ■ Pinned in Debugger	25 26 % 27 28 29	<pre> EmitSignal(SignalName.C EmitSignal(SignalName.C EmitSignal(SignalName.C } </pre>
		ሻወ	1 F

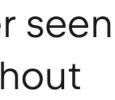


Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante, Unity CTO & Founder

ഴ





+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

+ Supports cutting edge Unity features like ECS, Burst

+ Shader support + Performance warnings as part of IntelliSense warnings

+ Integration with the Unity Editor and Unity Documentation

+ Recommended by one of the cofounders of Unity

+ Super familiar if you've used IntelliJ

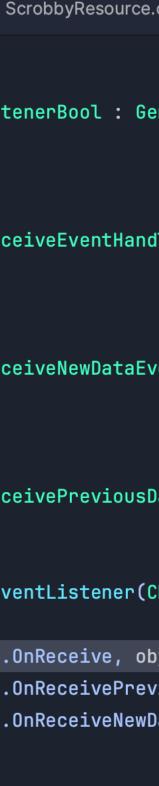
	\bullet \leftarrow \rightarrow S Sprout \checkmark $^{\circ}$ master \checkmark		
	Solution \checkmark	dModulo.cs	c# ScrobbyDependency.cs C# S
4	 Sprout · 3 projects C# Scrobby 	12	⊠ 13 usages 🛛 & Val Knight
00	 > & Dependencies > Generated > C Scrobby 	13 îo 14	<pre>public partial class EventListe {</pre>
	 > EventInvoking > EventInvokingData 	15	[Signal] & Val Knight
	 EventListening C# EventListenerBool.cs C# EventListenerDouble.cs C# EventListenerFloat.cs C# EventListenerGodotCollectionsArray.cs C# EventListenerGodotCollectionsArraybo 	19 20 21	public delegate void OnRece [Signal] & Val Knight public delegate void OnRece [Signal]
	C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerGodotCollectionsArrayGo C# EventListenerCodotCollectionsArrayGo Bookmarks	22 23	La Knight public delegate void OnRece I+4 usages
		24 ^ 25	protected override void Eve {
	Sprout ■ Pinned in Debugger	25 26 % 27 28 29	<pre> EmitSignal(SignalName.C EmitSignal(SignalName.C EmitSignal(SignalName.C } </pre>
		ሻወ	1 F

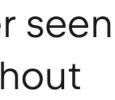


Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante, Unity CTO & Founder

ഴ

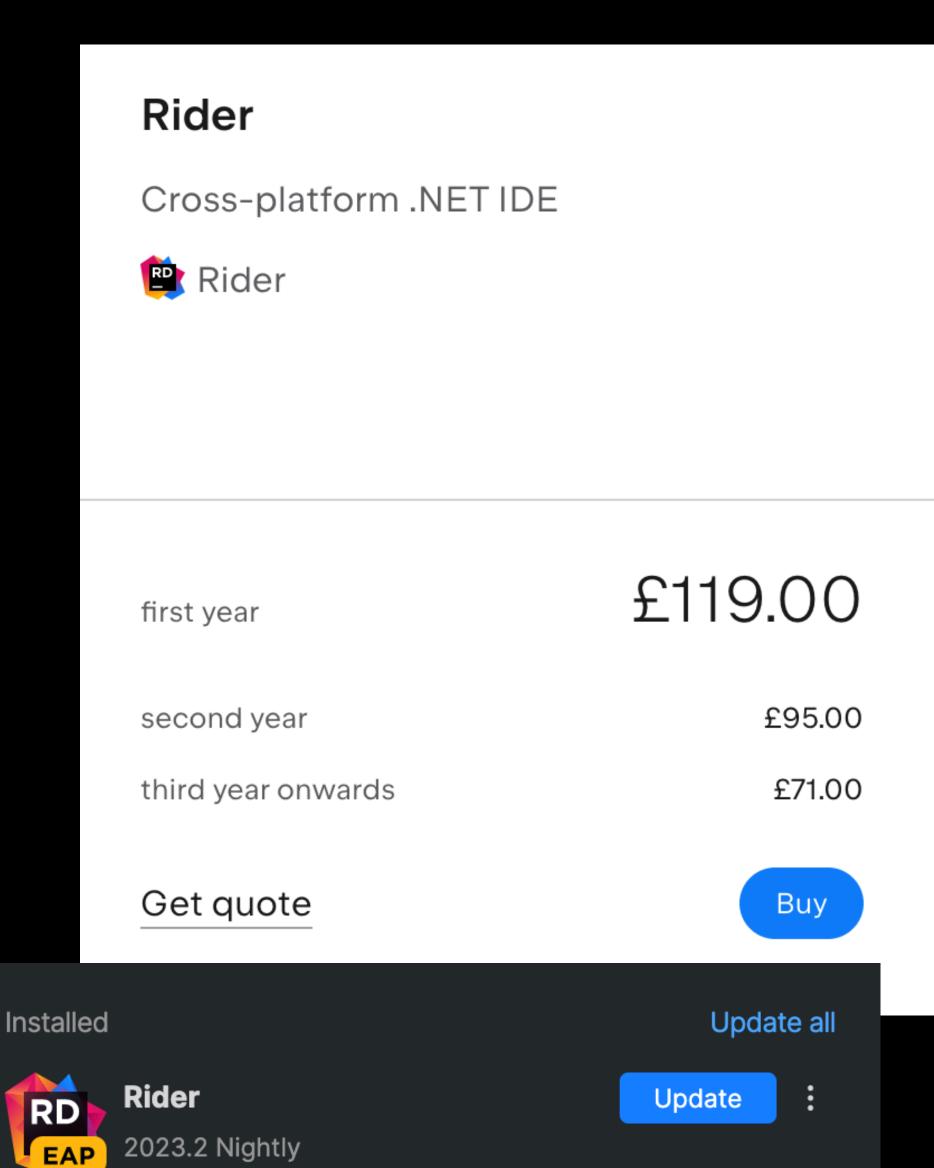




- Pretty memory intensive (but, so is VS Code...)

Rider



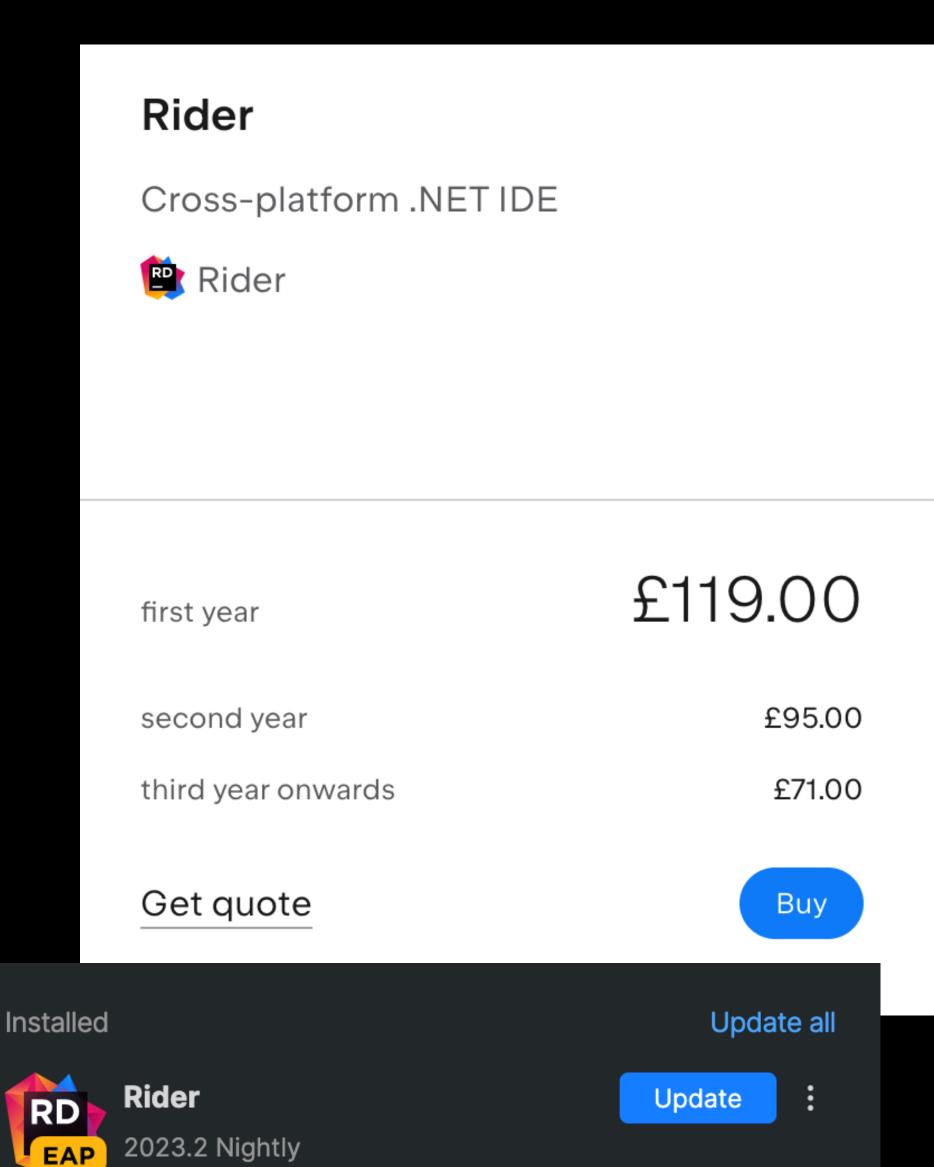


- Pretty memory intensive (but, so is VS Code...)

- Stable release requires signing up to JetBrains Student Program

Rider





- Pretty memory intensive (but, so is VS Code...)

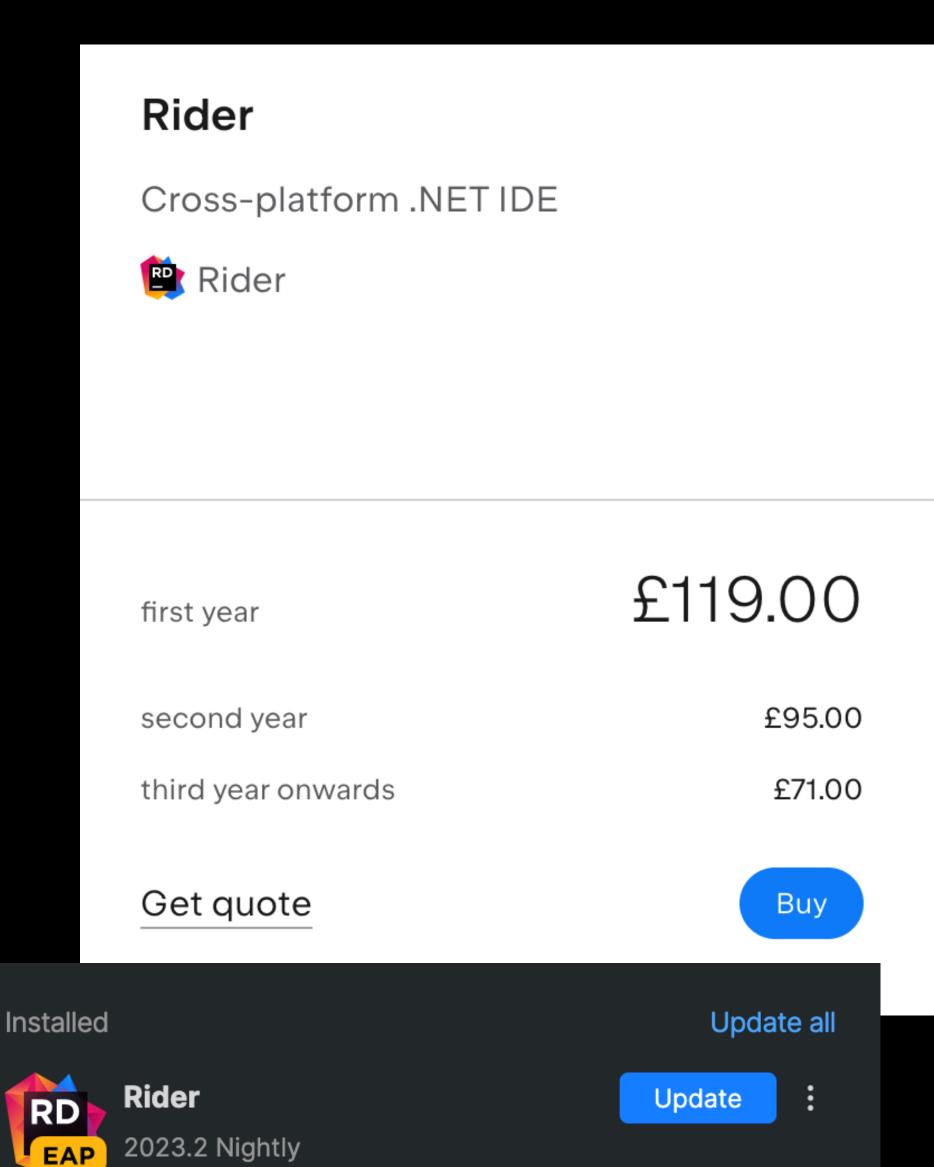
- Stable release requires signing up to JetBrains Student Program

- Post graduation you (or your employer) will have to pay*

- Rider Early Access / Nightly is completely free, but can have bugs

Rider





- Pretty memory intensive (but, so is VS Code...)

- Stable release requires signing up to JetBrains Student Program

- Post graduation you (or your employer) will have to pay*

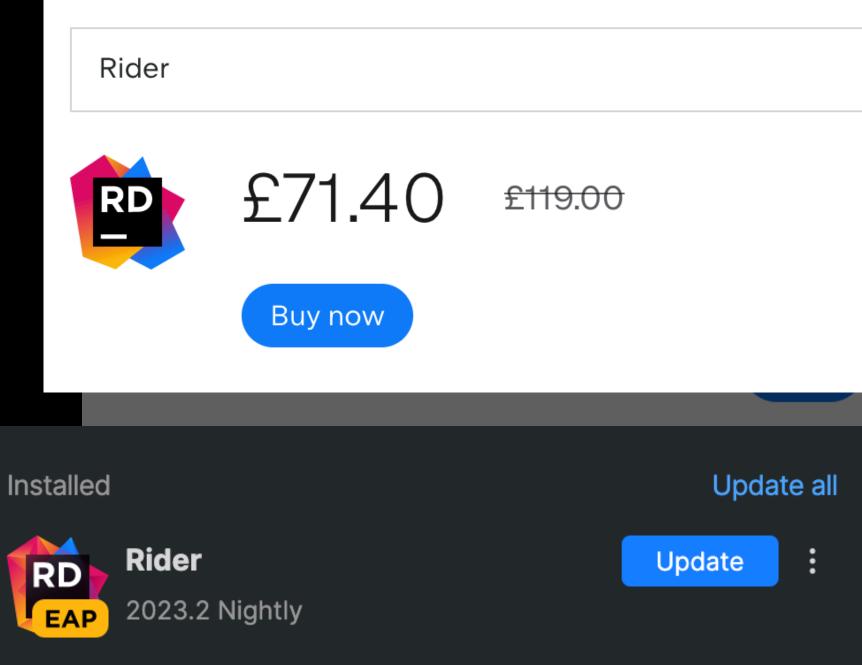
- Rider Early Access / Nightly is completely free, but can have bugs

- There is also a pretty heavy graduate discount, and you get to keep the version of Rider you pay for as a perpetual license



For graduates

Check out the 40% graduation discount for former Student I holders! Select the product of your choice, and the discount be applied automatically:





Visual Studio Windows

Visual Studio for Mac macOS

Visual Studio Code Windows, macOS, Linux

JetBrains Rider Windows, macOS, Linux

Visual Studio Windows

Windows devs: choose either

JetBrains Rider Windows, macOS, Linux

macOS / Linux: just use Rider

C# Fundamental 1: Value vs Reference Types

Value Type

- *Directly* contains it's data
- Generally simple primitives (floats, ints, bools, *not strings!*)
- Cannot be null!
- Passing it to another method creates a copy of the data to work

with unless you specify it's a reference with the "ref" keyword

Stored on the stack, meaning fast allocation & deallocation

Value lype

- *Directly* contains it's data
- Generally simple primitives (floats, ints, bools, *not strings!*)
- Cannot be null!
- Passing it to another method creates a copy of the data to work

with unless you specify it's a reference with the "ref" keyword

- Stored on the stack, meaning fast allocation & deallocation
 - she can explain if you're interested :)
 - Time constraints my beloved

Don't worry if you don't know what the stack means right now! Feel free to grab Val Mail and

Reference lype

- All your objects (strings, arrays lists, dictionaries, classes you define...)
- Passing it to another method is just referencing the same underlying data
- Reference to data existing in the "managed heap"
 - Again, feel free to ask Val what this means!
- Stored on the heap, with references existing on the stack
 - New references to existing data are still fast!
 - Creating data is relatively slow :(
 - Can lead to garbage, causing really bad performance issues : (((),

a quick game





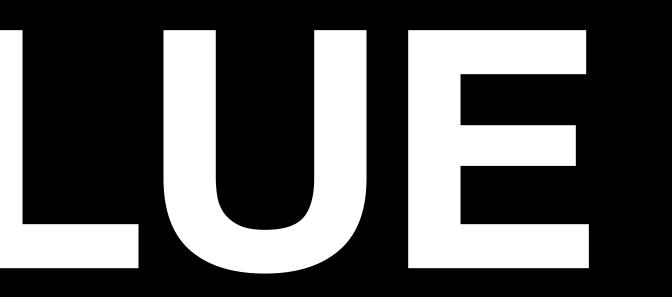
REFERENCE

Or

int someInt;



int someInt;



string someString;

REFERENCE

string someString;

float someInt;





float someInt;

string[] arrayOfStrings;

REFERENCE

string[] arrayOfStrings;

int[] myIntArray;

REFERENCE

int[] myIntArray;

Arrays of value types are still reference types

CoolCustomClass myObject; public class CoolCustomClass Ł public int x; public int y; }

CoolCustomClass myObject;

public int x;

public class CoolCustomClass

public int y;

}

CoolCustomStruct myStruct;

public struct CoolCustomStruct

public int x; public int y;



CoolCustomStruct myStruct;



public struct CoolCustomStruct

public int x; public int y;

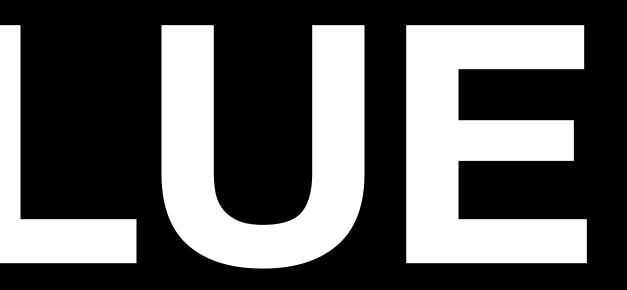
StructOfStrings myStruct; public struct StructOfStrings Ł public string x; public string y; }



}

StructOfStrings myStruct;

public struct StructOfStrings



public string x; public string y;

but x and y are still reference types

StructOfStrings myStruct;

public struct StructOfStrings

public string x; public string y;

Structs

- Great for small collections of other fields
- Can support methods, same as a class
- Structs are a value typed collection of fields
 - As such, passing a struct to another function will create a *copy*, not a reference by default!
 - You *can* pass a struct by reference with the *ref* keyword ask Val, and she can show you demos (time constraints!)



C# Fundamental 2: Garbage Collection

What is garbage?

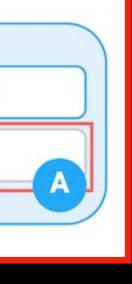
A reference type, with no references to it

What is garbage collection?

Your C# runtime going 👀 Looking 👀, specifically for objects in the managed heap with no references on the stack

int[] array	int[] array	int[]	int[] array
List	float[] array string	string	

Diagram of the Managed Heap Copyright Unity Technologies, 2023





Why is garbage bad?

- Garbage Collection is a very expensive operation to run, so we want to avoid generating garbage where possible
- It's expensive to allocate managed objects

What do you mean by "expensive"?

- If your game is targeting 60FPS, you have 16.67ms to get a frame out to the display
- Garbage Collection takes a lot of time to complete, especially if you have a lot of garbage
- This could cause frequent stuttering!

How do I avoid generating garbage?

- Keep an eye out for when you're constantly making new objects
 - Could it be done with a value type?
 - Could you modify the values of an object you allocate once?
- Watch out for strings and arrays!
 - Strings are references types; as such, generating and passing around strings can lead to suboptimal performance
 - This includes string concatenations (IE: "foo" + "bar" will generate garbage) Consider alternatives, such as passing an enum instead

- No!

- No!
- But it's useful to have an under where you can

- But it's useful to have an understanding, and get performance gains

- No!

- But it's useful to have an under where you can
- Unity's "Incremental GC" won' GC "spikes"

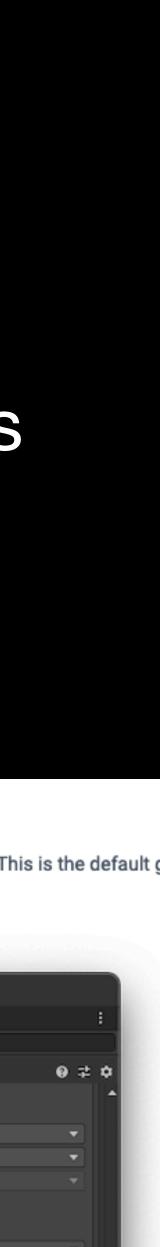
But it's useful to have an understanding, and get performance gains

- Unity's "Incremental GC" won't save you, but, it can help to reduce

Incremental garbage collection

Incremental garbage collection (GC) spreads out the process of garbage collection over multiple frames. This is collection behavior in Unity.

•••	Project Settings		
Project Settings			
		٩	
Adaptive Performance Audio Editor	Configuration	_	
Graphics	Scripting Backend	Mono	
Input Manager Memory Settings	Api Compatibility Level*	.NET Standard 2.0	
Package Manager	C++ Compiler Configuration	Release	
Physics	Use incremental GC		
Physics 2D Player	Assembly Version Validation	~	
i iajei			



- No!

- where you can
- GC "spikes"
 - PAL will likely be doing a hands-on profiling workshop!
 - Keep an eye out 👀

But it's useful to have an understanding, and get performance gains

- Unity's "Incremental GC" won't save you, but, it can help to reduce

Incremental garbage collection

Incremental garbage collection (GC) spreads out the process of garbage collection over multiple collection behavior in Unity

•••	Project Settings	Project Settings	
Project Settings			
Adaptive Performance Audio Editor	Player	-	
Graphics Input Manager	Configuration Scripting Backend	Mono	
Memory Settings Package Manager	Api Compatibility Level* C++ Compiler Configuration	.NET S Releas	
Physics Physics 2D	Use incremental GC		
Player	Assembly Version Validation	~	

andard 2.0

C# Fundamental 3: Properties

Java Land getters, setters? more like get-out-of-town-ers. gottem

public class Player {
 private float playerHealth;

public float GetPlayerHealth() {
 return playerHealth;
}

public void SetPlayerHealth(float f) {
 playerHealth = f;



public class Player { private float playerHealth;

public float GetPlayerHealth() { return playerHealth; }

public void SetPlayerHealth(float f) { playerHealth = f;



public class Player { public float playerHealth; }



public class Player { public float PlayerHealth; }

* naming conventions

but what about encapsulation???



C# Land the cool part

```
public class Player
{
    public float PlayerHealth;
}
```

C# Land the col part

```
public class Player
    private float _playerHP;
    public float PlayerHealth
        get
            return _playerHP;
        }
        set
        {
            _playerHP = value;
        ł
        J
```

C# Land the col part

```
public class Player
    private float _playerHP;
    public float PlayerHealth
        get
            return _playerHP;
        set
        {
            _playerHP = value;
        J
```

What used to be just a value has now been changed into a free getter and setter

C# Land the col part

```
public class Player
    private float _playerHP;
    public float PlayerHealth
        get
            return _playerHP;
        set
            _playerHP = value;
```

What used to be just a value has now been changed into a free getter and setter

This means you do not need to write, generate or otherwise include getters and setters!

C# Land the col part

```
public class Player
    private float _playerHP;
    public float PlayerHealth
        get
            return _playerHP;
        set
            _playerHP = value;
```

What used to be just a value has now been changed into a free getter and setter

This means you do not need to write, generate or otherwise include getters and setters!

This "fake" field is called a "property"!



C# Fundamental 4: Delegates

What is a calback?

- A callback is some method you provide to another method, to run when Something happens
- e.g. downloading files
 - you have a function called ProcessText(string text);
 - you have a function called DownloadText
 - you WANT to tell DownloadText to call ProcessText when it's done
 - But, you don't want to hard-code this

Callbacks in Java

- SO! Sometimes, we need callbacks!
- We could do something with an interface

Interface Based Callbacks in Java

```
interface ICallback<T> {
    public void CallbackMethod(T data);
}
class Foo {
    public void DoAction(ICallback<Integer> c) {
        // this is doing complex maths
        int randomNumber = 4;
        c.CallbackMethod(randomNumber);
public class Bar implements ICallback<Integer> {
    public static void main(String[] args) {
        Foo f = new Foo();
        Bar b = new Bar();
        // this will allow Foo to call our callback!
        f.DoAction(b);
    }
    public void CallbackMethod(Integer data) {
        System.out.println(data);
}
```

This uses generics! but; don't super worry if generics aren't your jam just focus on how ICallback is used!

Interface Based Callbacks in C#

```
interface ICallback<T> {
    public void CallbackMethod(T data);
}
class Foo {
    public void DoAction(ICallback<int> c) {
        // this is doing complex maths
        int randomNumber = 4;
        c.CallbackMethod(randomNumber);
public class Bar: ICallback<int> {
    public static void main(String[] args) {
        Foo f = new Foo();
        Bar b = new Bar();
        // this will allow Foo to call our callback!
        f.DoAction(b);
    }
    public void CallbackMethod(int data) {
        Debug.Log(data);
}
```

This uses generics! but; don't super worry if generics aren't your jam just focus on how ICallback is used!

Calbacks in Java

- Sometimes, we need callbacks!
- We could do something with an interface
- But this isn't very flexible
 - Everything needs to inherit the interface

Very rigid, and can't dynamically decide what's the callback at runtime

Callbacks in C#

- Sometimes, we need callbacks!
- We could do something with an interface
- But this isn't very flexible
 - Everything needs to inherit the interface
- Maybe C# has a better way...

Very rigid, and can't dynamically decide what's the callback at runtime

Interface Based Callbacks in C#

```
interface ICallback<T> {
    public void CallbackMethod(T data);
}
class Foo {
    public void DoAction(ICallback<int> c) {
        // this is doing complex maths
        int randomNumber = 4;
        c.CallbackMethod(randomNumber);
public class Bar: ICallback<int> {
    public static void main(String[] args) {
        Foo f = new Foo();
        Bar b = new Bar();
        // this will allow Foo to call our callback!
        f.DoAction(b);
    }
    public void CallbackMethod(int data) {
        Debug.Log(data);
}
```

Delegate Based Callbacks in C#

```
class Foo {
    public delegate void SetTextCallback(int data);
    public void DoAction(SetTextCallback cb) {
        // this is doing complex maths
        int randomNumber = 4;
        cb(randomNumber);
public class Bar {
    public static void main(String[] args) {
        Foo f = new Foo();
        Bar b = new Bar();
        // Explicitly send CallbackMethod!
        f.DoAction(b.CallbackMethod);
    }
    public void CallbackMethod(int data) {
        Debug.Log(data);
```

Callbacks in C#

- Sometimes, we need callbacks!
- We could do something with an interface
- But this isn't very flexible
 - Everything needs to inherit the interface
- Maybe C# has a better way...
- delegates for callbacks, delegates in dictionaries, etc etc etc...)

• As a delegate is a datatype, we can do more complex things (e.g., lists of



Example use of delegates

- How would you build an achievement system?
- Do you check all your achievements, every frame?
 - No!
- Instead, lets have other bits of the game simply tell the achievements when things happen...
- E.g., when the player moves, the achievement system will just have a list of delegates to run!

Example use of delegates

- How would you build an achievement system?
- Do you check all your achievements, every frame?
 - No!
- Instead, lets have other bits of the game simply tell the achievements when things happen...
- E.g., when the player moves, the achievement system will just have a list of delegates to run!
 - We have just come up with the **Observer Programming Pattern**



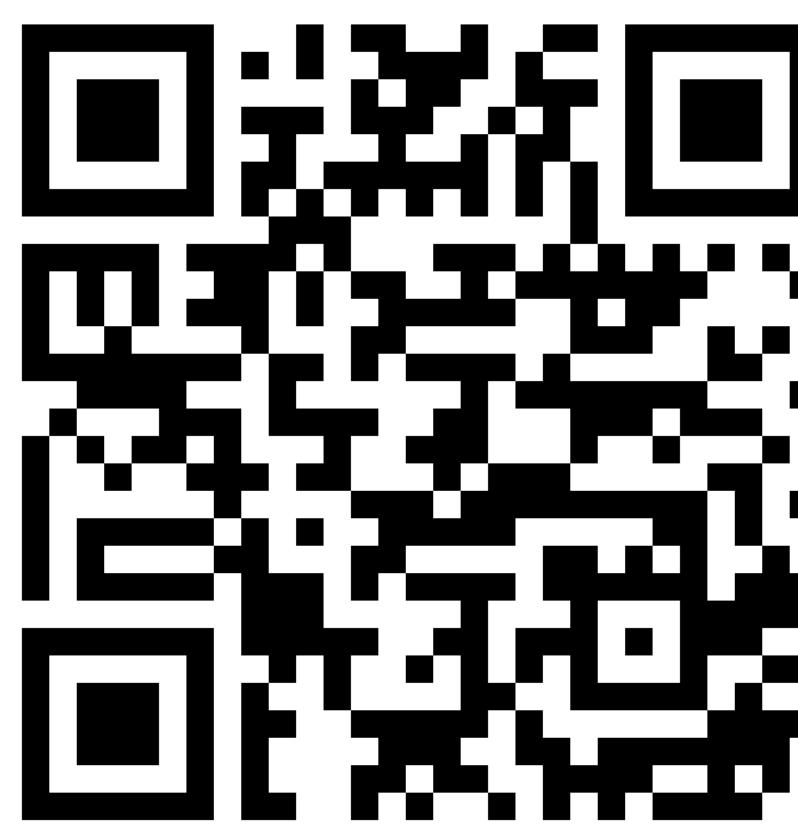
lakeaways

- Pick an IDE Visual Studio or JetBrains Rider are recommended!
- Keep in mind whether something is a value, or a reference!
- Be aware of garbage, and try to minimise generating garbage in your code!
- Delegates can really help clean up your code!

Further reading

Further reading

- This QR code should take you to a bunch of links for further reading
 - Outside of this, also explore learn.microsoft.com, Unity Learn, and the Unity Documentation!
 - Microsoft's & Unity's documentation are both really good, especially compared to what you may be used to
- It also has a direct link to the page to get the JetBrains student plan, for free Rider!





Further reading / Some highlights!

- Unity Learn
 - Great for beginners, and is usually up to date and trusted
 - (love u Unity Learn team <3) ightarrow
- Catlike Coding
 - Great resource for when you want to go a bit deeper! \bullet
- Acerola

Amazing channel if you ever want to explore the more shadery / technical art side of gamedev - we'll likely be running some shader workshops in future, but, well worth a watch!



Untitled 4 — Edited

hey guys welcome to my C# tuturial, today we will be learning how to make .nET applications





- Be careful with YouTube tutorials (or, StackOverflow, Unity Forums/Discussions, etc etc...) and using ChatGPT
- Unity has changed quite a bit in the past few years
- Al tools are primarily trained on hobbyist code there aren't many open source commercial Unity games! - so, common mistakes will get repeated, and it will happily spout misinformation
- If you need advice on if a tutorial should be trusted, reach out to us on PAL! We can take a quick skim over, and give a temperature check for you







Next Sessions

Next Sessions

- Thursday
 - tips, remaking a certain cult-classic in the Informatics department

- Other upcoming sessions
 - Will be announced on the Englnf Discord!
 - PAL can also help out with specific gamedev queries you may have!

Live game making workshop! Come along, and learn lots of practical Unity

Thursday 12th, **FTL Lab**, **14:00 -> 15:00**

// [Slide left intentionally empty for Q&A]