QA MASTERCLASS

About

About Me

- Hi, I’m Emerson!
- I’ve been developing games for nearly a decade.
  - Singleplayer & Multiplayer
  - Unity & Unreal
  - Console, VR/XR, WebGL, Desktop, Mobile
- I’ve been creating world-first tech in the VR/XR since 2019

About Me

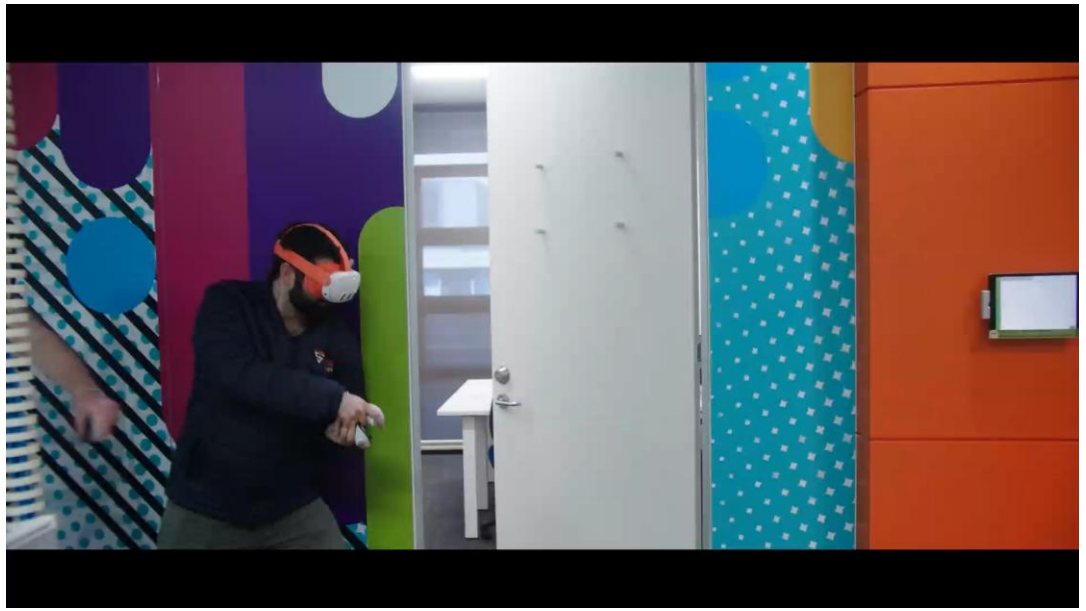
- In 2019, I created the world's first public implementation of co-location on the Quest 1
- The first game to allow multiple people in the same space to battle each other at the same time.
- TritonVR (2019-2022)
  - 155k downloads
  - 350k views
  - One of the top games on SidequestVR.com

About Me

- I joined the OperatorXR team in 2022.
- OperatorXR is a co-located use-of-force training simulator for police and military personnel.
- Now in use by 50+ agencies all around the world!

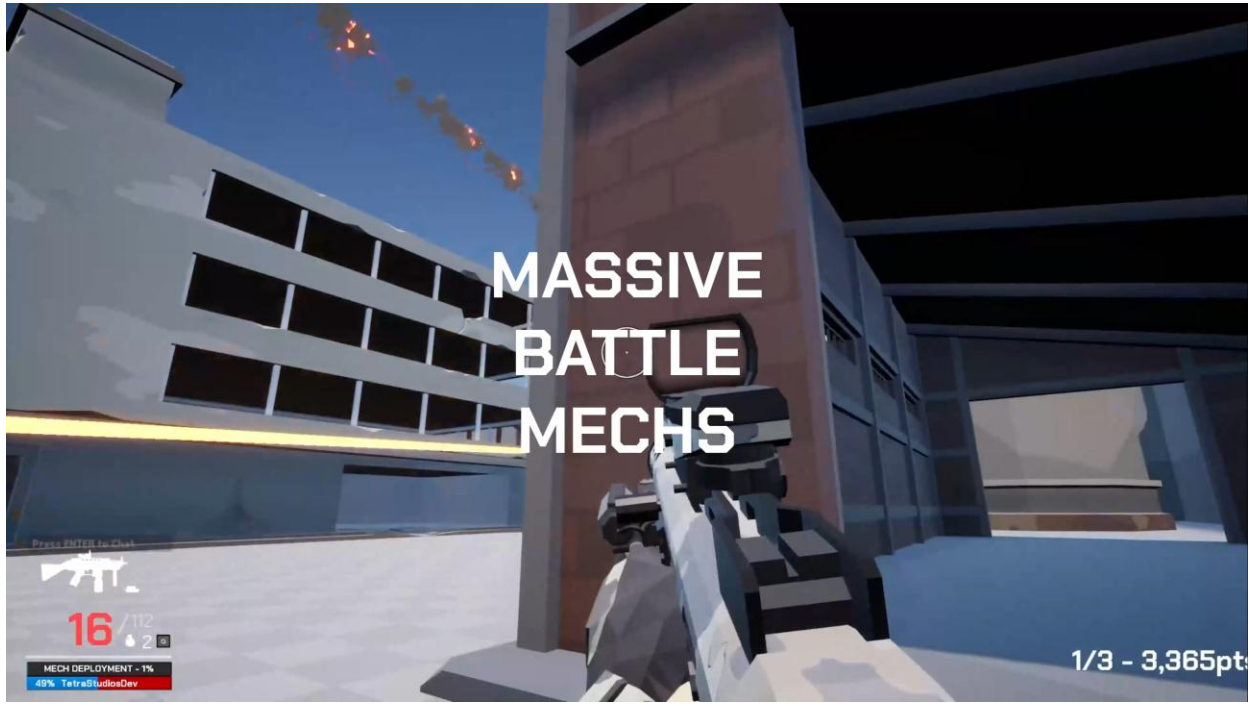
About Me

- Worked with Meta on TritonXR between 2023-2025.
- An XR FPS that pushed the boundaries of the Meta Quest platform.
- Introduced 3 world-first XR innovations!



About Me

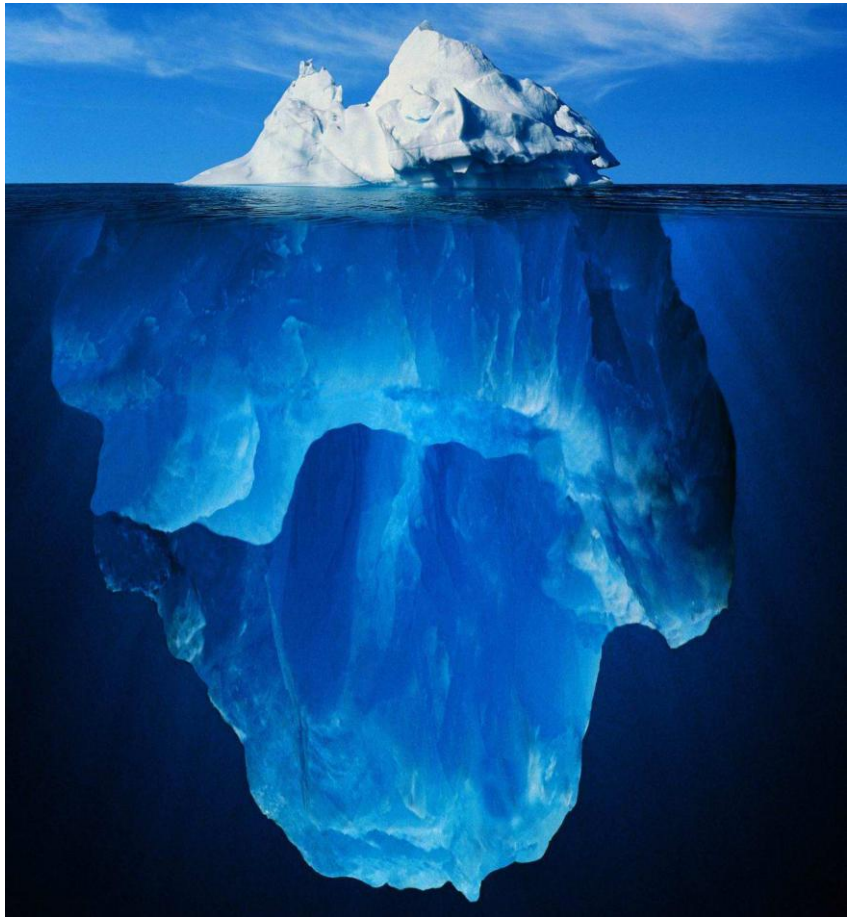
- Now taking lessons from TritonXR and adapting it to Project Shatterpoint
  - Titanfall's Core Gameplay
  - The Final's Destruction
  - Halo's Social Multiplayer
- Within 3 months of starting development:
  - 2 million views
  - Coverage by Yahoo News, Creative Bloq and 80.lv
  - Demo available now!



Talk Structure

1. Basics of Bugs
2. More Than Technical
3. Reading Between The Lines
4. Putting It All Together

Talk Structure



Part 1: The Basics of Bugs

What is a bug?

What is a bug?

Anything that doesn’t work as the developer intended

Defining Bugs: Technical Bugs

- The standard “it doesn’t work” bug.
- Caused by code
- Although usually the fault of the developer, the engine, network latency and the platform the player is using also can cause this.

Technical Bug Example: Moonwalking Horse

Why did this bug happen?

- Horse is normal.
- The horse rotates 180 degrees randomly.
- The movement system moves the horse forwards up the hill, and plays the “walk_forward” animation.
- As the horse is the wrong way around, the horse’s movement and animation are now misaligned

How to fix this?

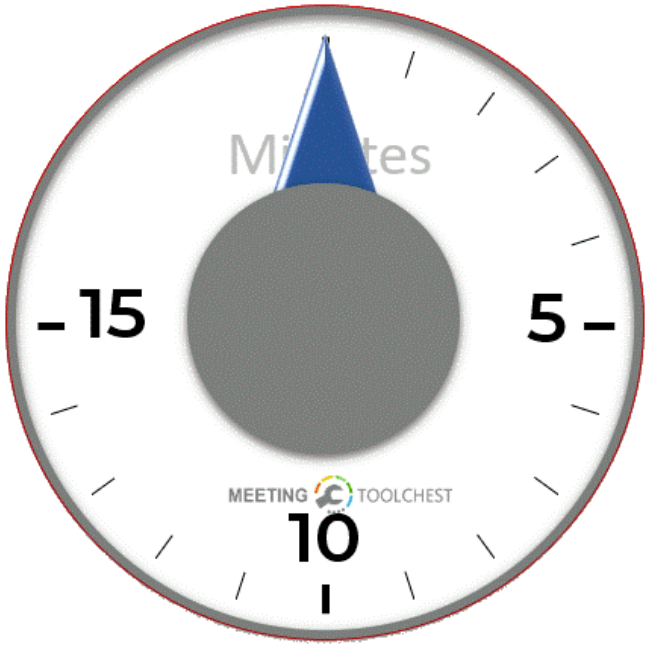
- Quick Fix: Ensure horse rotation always follows the direction of movement for walking.
  - Quick, but can introduce new problems (what direction should the horse be when falling?)
- Long-term Fix: Figure out why the horse decided to turn 180 degrees in the first place.
  - May be due to level geometry
  - Fixing this means keeping the current system the same!

Lessons the Moonwalking Horsie can teach us!

- State switching (player on horse -> player off horse) is usually fragile, and can cause issues if you’re not careful.
- Test your movement on your actual level geometry, not in a separate test scene.
- AI Navigation has a lot of edge-cases like this – try to have ways to deal with it

Download!

1. Visit: github.com/TetraStudios/UTSWorkshop-QAMasterclass
2. Open the UTSWorkshop-QAMasterclass.exe file under the “PackagedBuilds” folder.
3. Don’t open the Unity Project just yet! We will do this later ☺



15 mins

Try it yourself!

How many bugs can you find in the provided game?

Write down each one you come across.

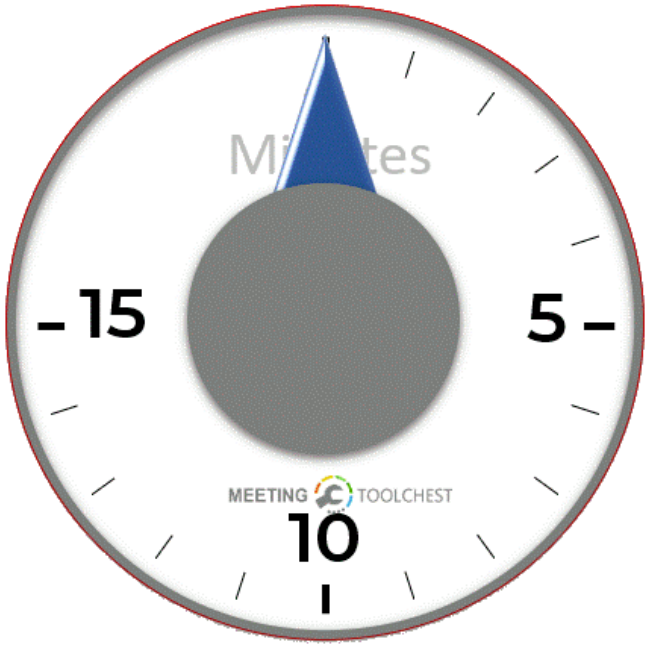
Remember, a bug is *“Anything that doesn’t work as the developer intended”*!

How many did you get?

There are 9 bugs to find!

My 5 Basic Methods To Find Bugs

1. Spam every key on the keyboard.
2. State switching (ie UI Fading, Horse -> Player, Level Changes) are often fragile – capitalise on this!
3. Spam the same action quickly.
    1. Includes Player actions and UI actions
4. “Speedrun” the level – see if you can complete it early by playing it as it was NOT intended.
5. Spam opposite actions quickly
    1. (ie Equip Item -> Drop Item -> Equip Item)



10 mins

Try it yourself!

Go back, and use these methods to find the other bugs you missed.

Bug Documentation

1. Summarise the bug
2. Note down reproduction steps!
    1. Videos and images are really helpful here
3. Label it with a priority of how impactful the bug is to the end user
    1. Visual quirk? Low Priority
    2. Player can’t spawn? Critical Priority

Add parent / SPT-115

1

Bolt-Actions don't work on high-lag environments

+ Add Apps

Description

When playing in a high-lag environment, there is a small chance that a sniper rifle will not cycle the bolt, preventing the player from shooting.

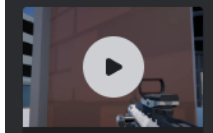
TO REPLICATE:

- Join a high-ping lobby
- Select the Sniper as your weapon
- Keep firing repeatedly - bug has a chance of occuring with each shot.

Environment

- Play in a packaged build
- The higher your ping, the more likely this is to occur

Attachments 1



2025-05-31 ... rim.mp4
23 Jun 2025, 08:59 AM

Activity

All | Comments | History | Work log | Checklist history

Add a comment...

Looks good! | Need help? | This is blocked... | Can you clarify...?

Pro tip: press M to comment

Emerson
8 seconds ago (edited)

This has been reported by a few players:

- Discord - Group Chat That's All Fun & Games
- Discord - Group Chat That's All Fun & Games

Need to make this a priority!

To Do

Details

Assignee	Emerson
Reporter	Emerson
Development	Open with VS Code Create branch Create commit
Story Points	0.5
Sprint	SPT Sprint 7 +2
Priority	High

More fields

Original estimate	0m
Time tracking	No time logged
Team	None
Fix versions	None
Affects versions	None
Parent	None

Automation Rule executions

Checklist Open Checklist

Clockify Start / Stop

Created May 26, 2025 at 12:52 PM
Updated in 6 seconds

Configure

Description of bug

Replication steps

Environment info

Video of bug happening

Bonus – link places where players have reported it

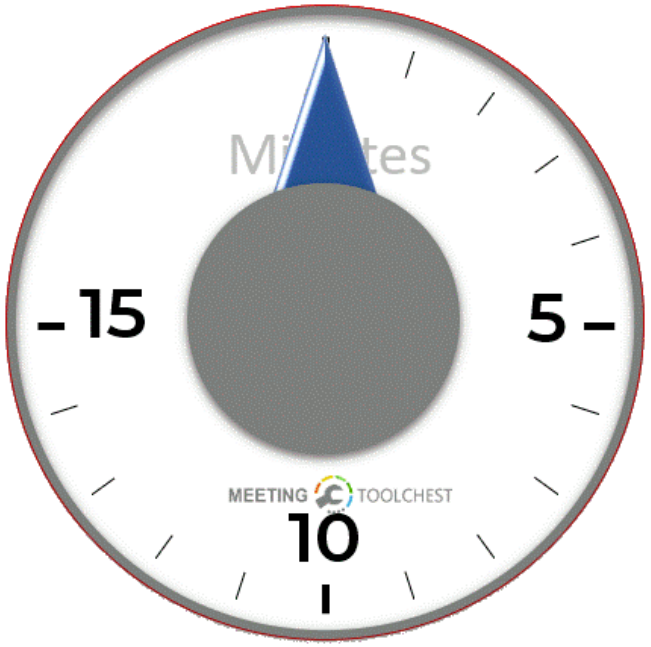
People responsible for fixing it

Code Commits

Story Points

Priority/Impact

Additional version information



10 mins

Try it yourself!

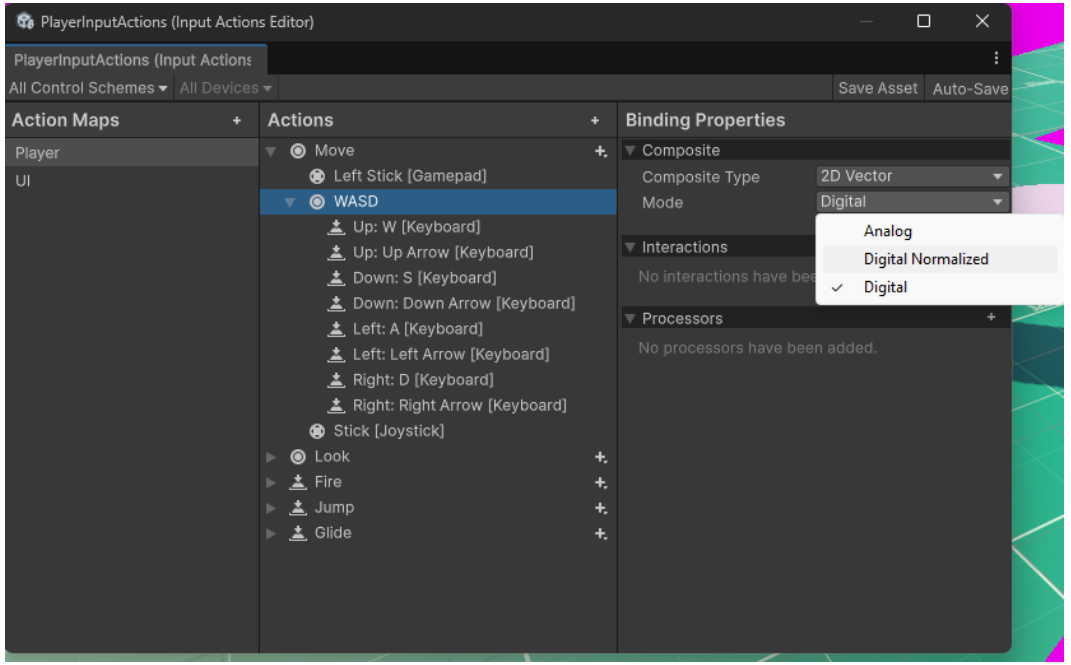
Go back, and ensure each bug has at least a summary, reproduction steps, and an impact.

All Technical Bugs

1. Input is not normalized
2. Menus do not get despawned
3. Menu will restart fading animation
4. No Killbox
5. Premature Culling
6. Multiple Win
7. Z-Fighting
8. Not all the killboxes work
9. High Score doesn’t work

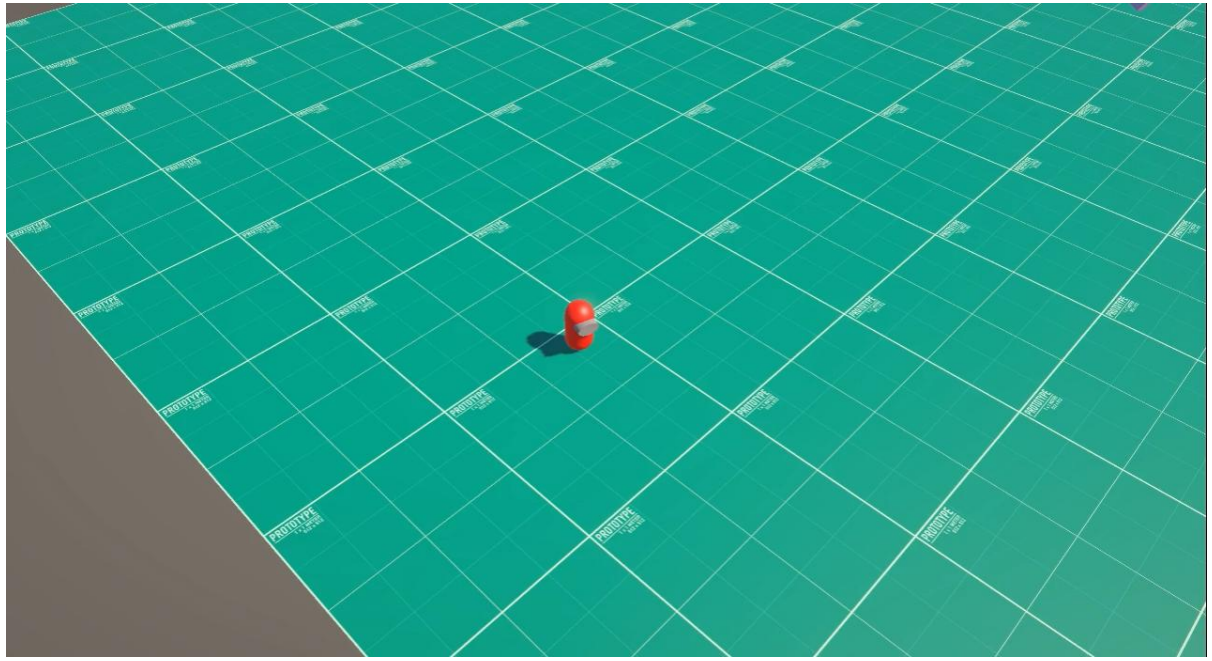
Bug 1: Normalizing Input

- Try holding A + D, and your character will move faster!
- This is because:
  - Up (Y = 1) + Right (X = 1) = 1 + 1 = 2
  - Sqrt(2) = 1.41, which provides a 41% speed boost compared to just moving in one direction
- To solve this:
  - Normalize the input (which makes the maximum direction value = 1)
    - inputVector.normalized;
  - Use “Digital Normalized” input in Unity’s Input Action system.



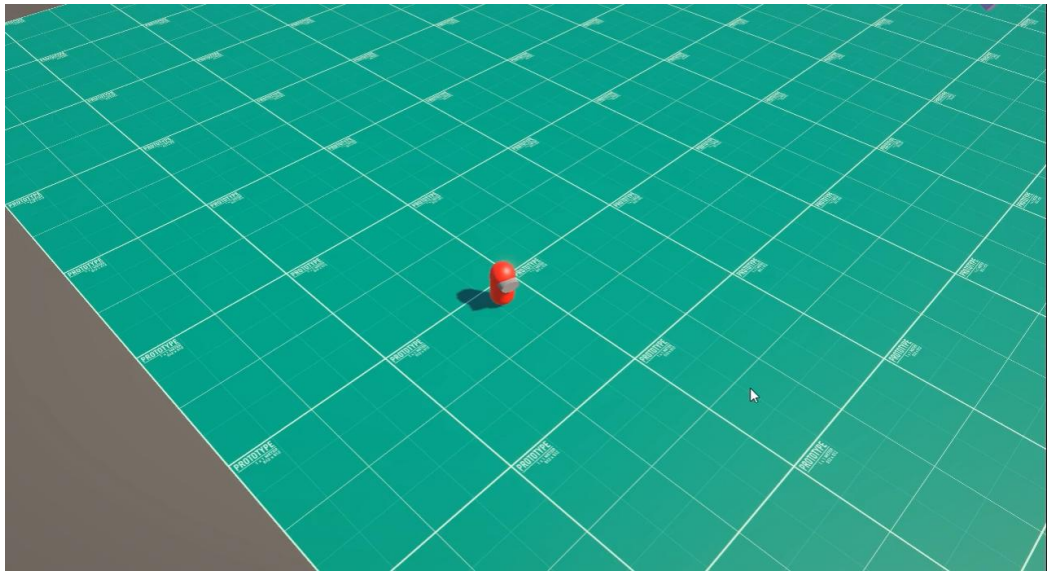
Bug 2: Menu does not get despawned

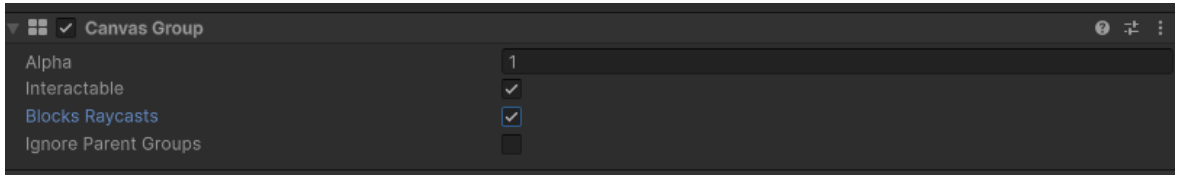
- The menu does not get hidden after pressing “resume”
- This is because we forgot to despawn it!
- This bug may be missed, because we don’t expect the menu to still be there, and therefore don’t test the behaviour again.
- LESSON: Never assume your game works until proven otherwise!



Bug 3: Menu will restart fade animation

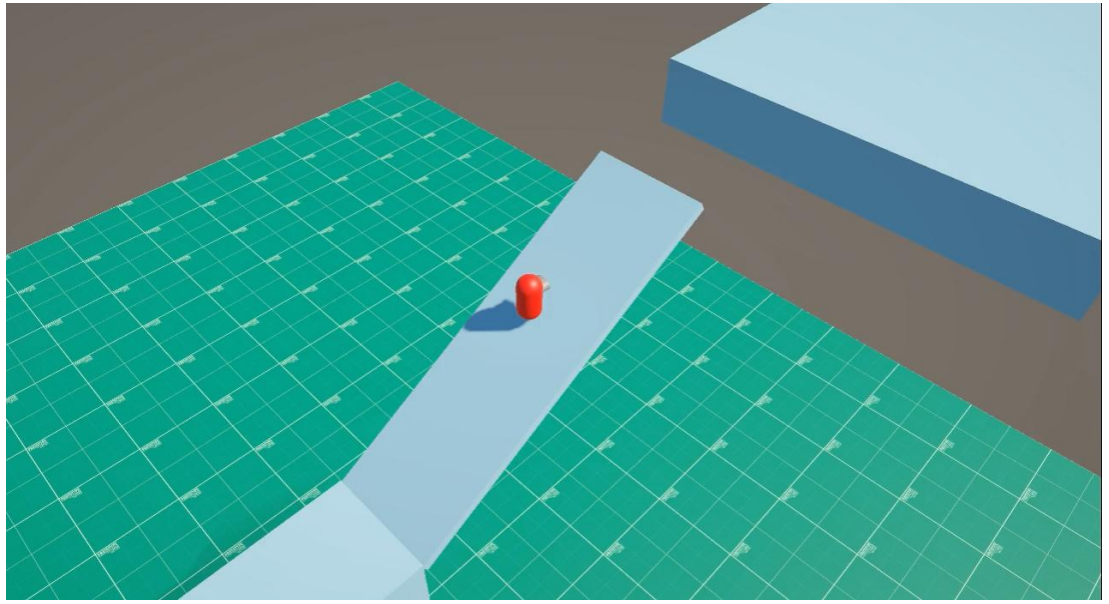
- Following on from bug 2, if you press the “resume” button while it is fading, you will restart the fading process
- To fix this, ensure that either:
  - Play the “fade in” animation starting from the current fade time.
  - Or ensure no other animations can be played.
  - Or make the menu non-interactable when it is being destroyed
    - Disable “Block Raycasts” to make it non-interactable





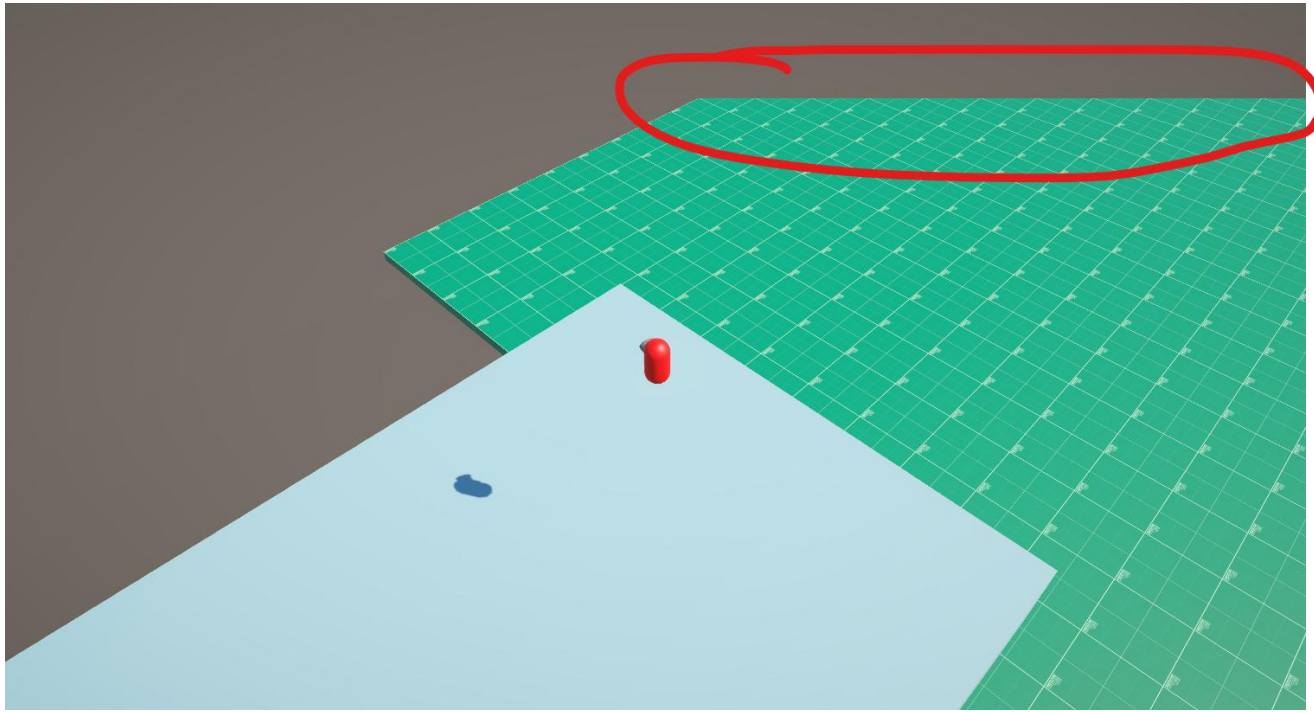
Bug 4: No Killbox

- Although obvious in this case, you should always have a way to reset the player if they ever get out of bounds.
- “Failing Gracefully!”



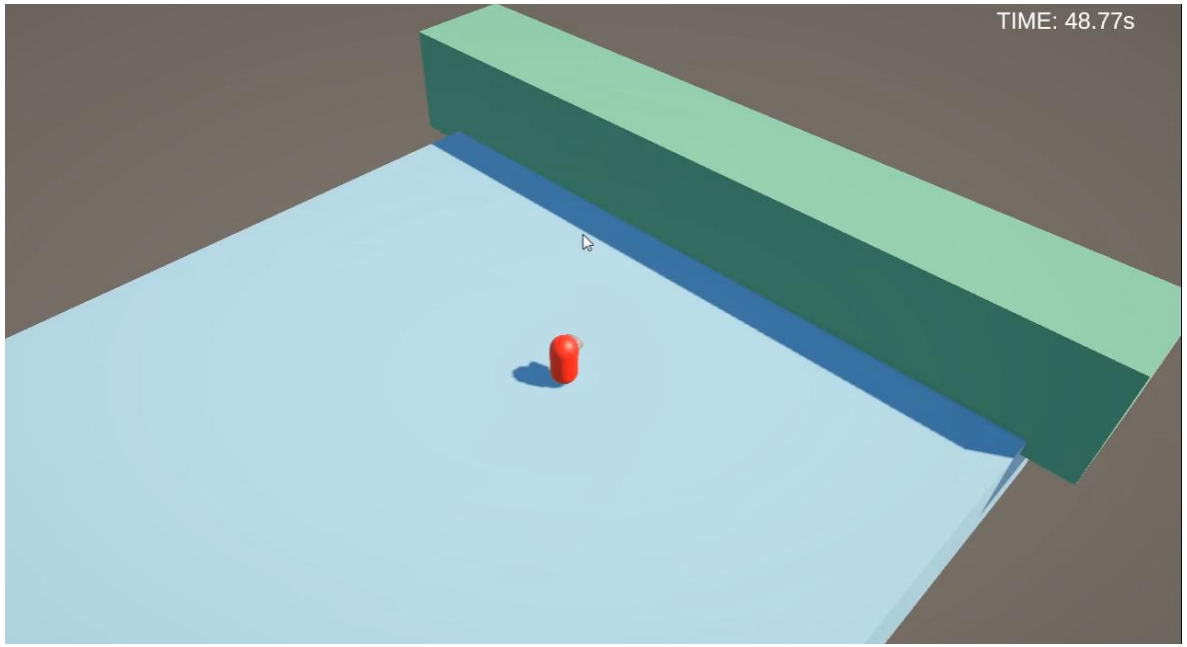
Bug 5: Premature Culling

- Camera culling can cause pop-in if the player goes somewhere you don’t expect!



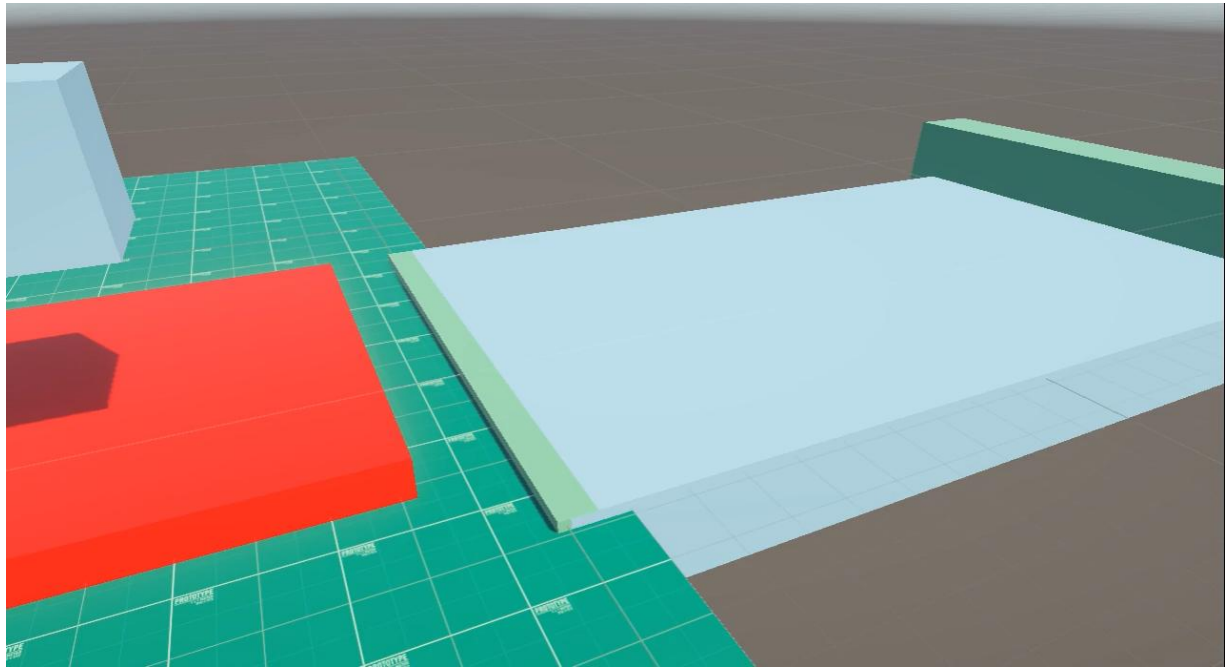
Bug 6: Multiple Wins

- If you run into the win box repeatedly, you can run the win condition multiple times!
- Again, a “do once” condition will fix this.



Bug 7: Z-Fighting

- Putting textures on the exact same Z level will cause Z-Fighting, seen here.
- Ensure you have the objects on different Z levels to fix this.



Bug 8: Not all the killboxes work!

- One of the killboxes does not restart the game.
- Find out which one it is!

Bug 9: High Score Doesn’t Work

- The HighScore system uses PlayerPrefs.
- We haven’t initialised it properly, so we can’t actually save a high score!
- If the default return value is 0, how can we ever be lower than that?

End of Part 1
Any Questions?

Part 2: More Than Technical

Defining Bugs: Design Bugs

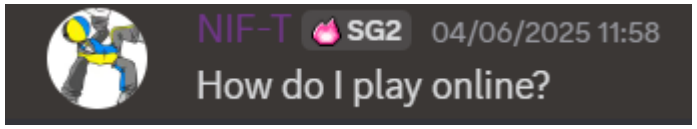
- A UX/UI issue where players don’t know how to execute a certain action.
- Caused by lack of player understanding
- Just as important as a technical bug, as it prevents players from experiencing the game’s content

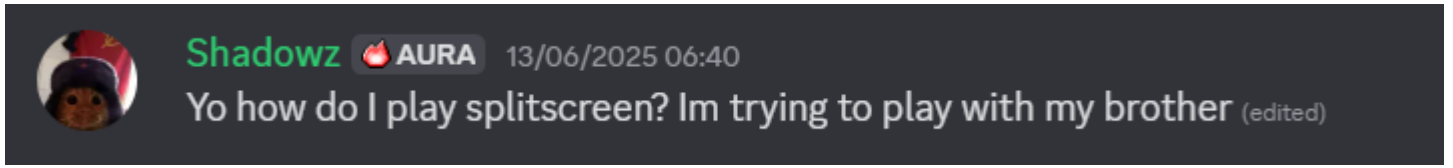
Design Bugs

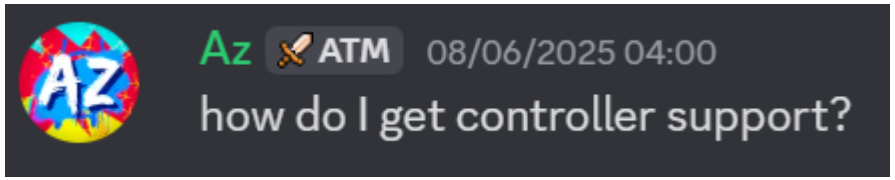
Almost always starts with “How do I…”

Examples of Design Bugs

Real Screenshots from the Tetra Studios Discord!







All of these features mentioned are 100% fully functional, but players are prevented from using them due to a lack of understanding!

Defining Bugs: Accessibility Bugs

- Similar to Design bugs, where players cannot experience game content because of their environment or a disability.
- Can be really hard to spot, so user testing with a wide range of people is really helpful here.

Defining Bugs: Accessibility Bugs

- Common Examples (from GameAccessibilityGuidelines.com)
  - Allow holding inputs rather than spamming inputs during Quick Time Events
  - Highlight important or key words
  - Ensure no essential key info is provided by colour alone
  - Captions
  - Rebindable Keys

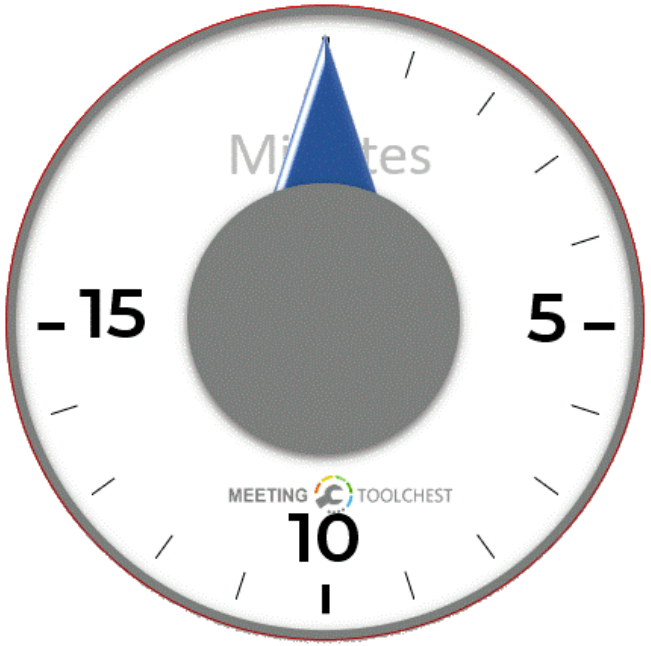
Accessibility – Curb Cut Effect

- Don’t disregard these sort of bugs!
- Although they may only negatively impact a few people, fixing them can be beneficial to many more players

Defining Bugs: Accessibility Bugs

- Common Examples (from GameAccessibilityGuidelines.com)
  - Allow holding inputs rather than spamming inputs during Quick Time Events
  - Highlight important or key words
  - Ensure no essential key info is provided by colour alone
  - Captions
  - Rebindable Keys

Defining Bugs: Accessibility Bugs



10 mins

Try it yourself!

Go back, and find any Design or Accessibility Bugs:

The features may work, but they may not not clear or be inaccessible to some people and can prevent players from experiencing the game.

All Design Bugs

1. Use Common Inputs
2. Tutorial is all front-loading
3. Able to Skip the Level
4. Telegraph Damage

Design Bug 1: Use Common Inputs

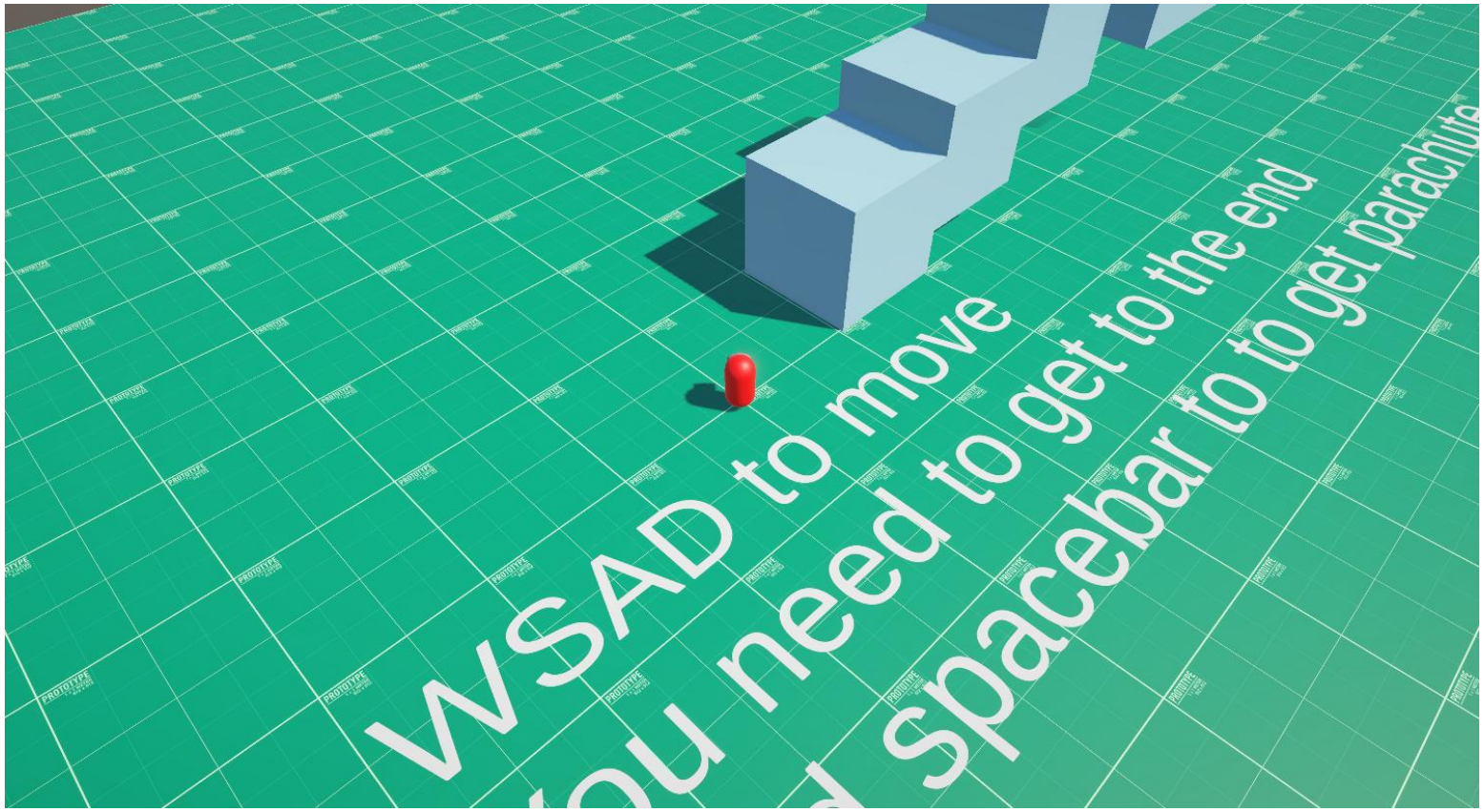
Escape creates menus, but doesn’t remove them.

Ensure you use the same input methods as other games use to reduce user friction

Design Bug 2: Tutorial Is Front-Loading

Front-loading tutorial info is really bad.

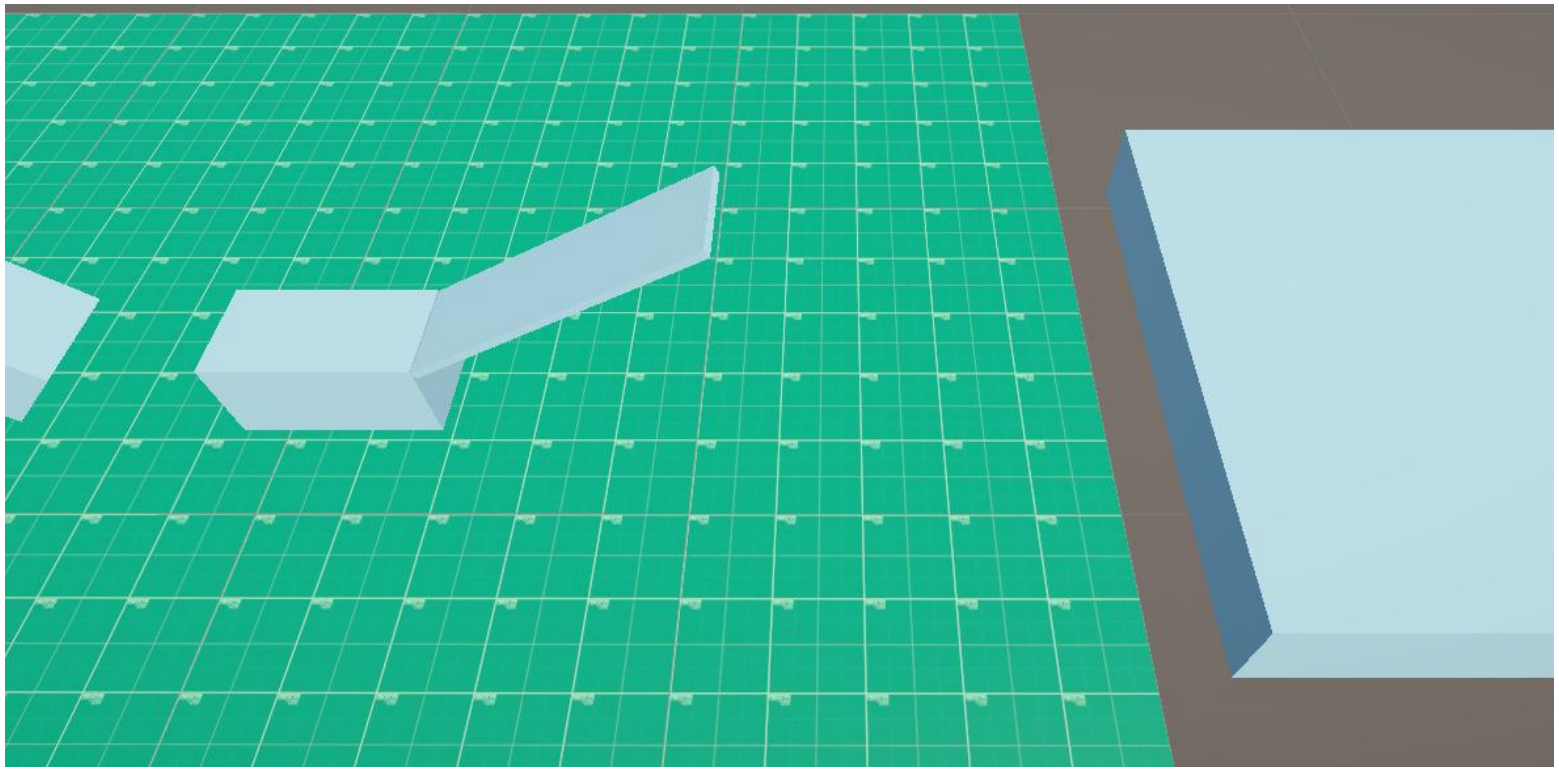
If you assume players have all the knowledge on how to play the game from this point, you WILL run into players getting stuck.



Design Bug 2: Tutorial Is Front-Loading

This section assumes the player knows how to use the parachute.

If they don’t know how to do this, they are stuck!

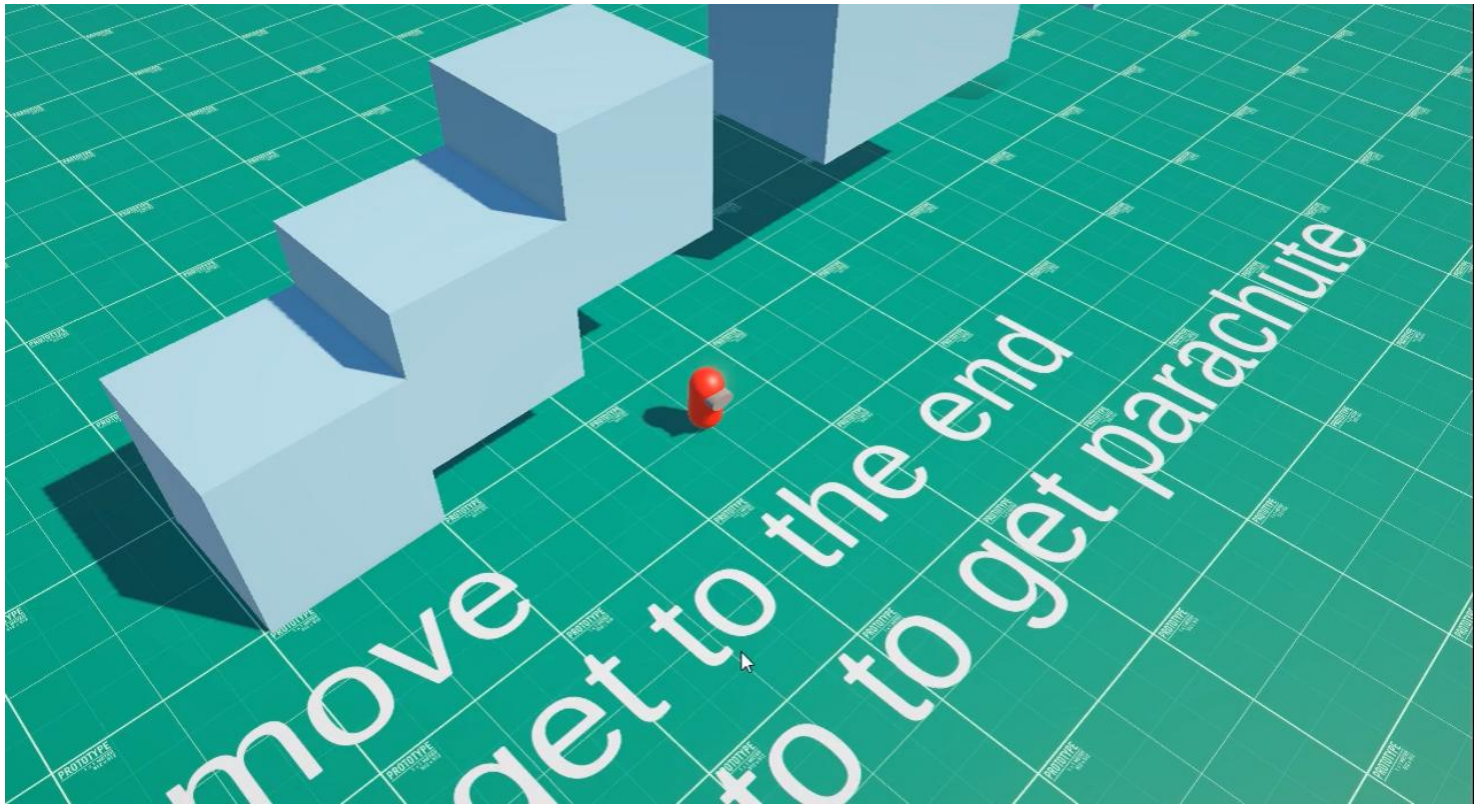


Design Bug 3: Easily Able To Skip

Don’t assume players will take the path you specifically make.

If you’re adjusting movement values, be sure to go through your level again!

PRO TIP: Spamming jump is a great way to test this.

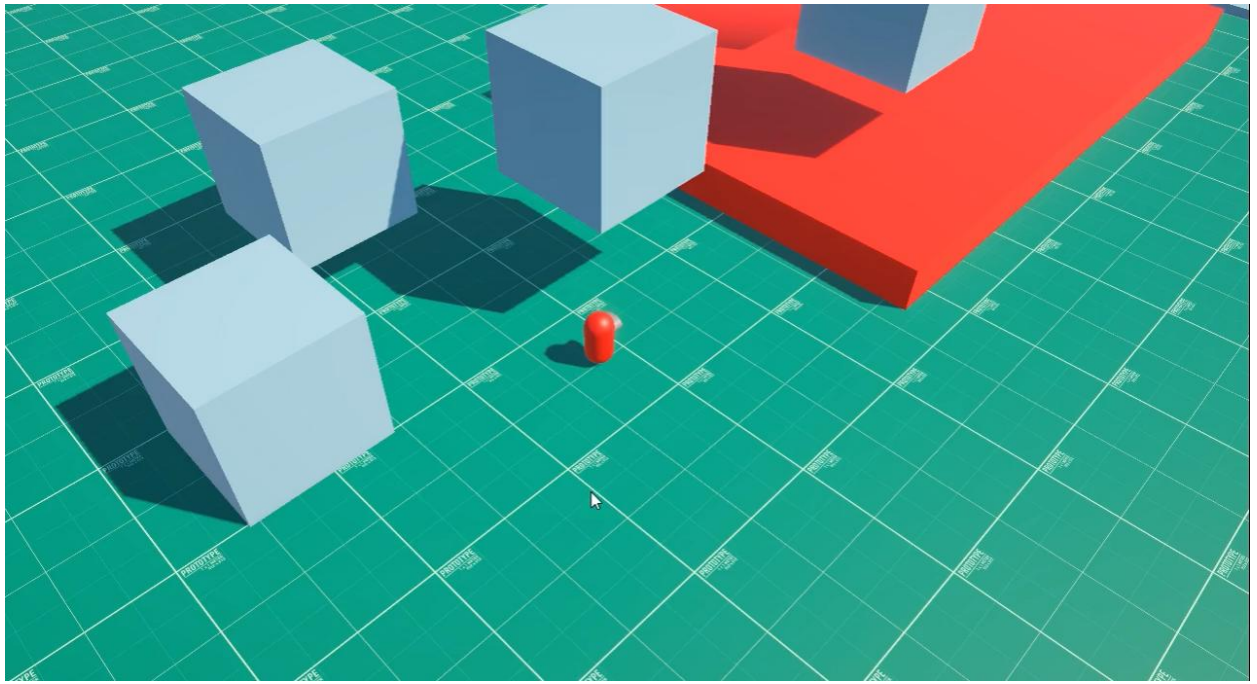


Design Bug 3: Easily Able To Skip

Don’t assume players will take the path you specifically make.

If you’re adjusting movement values, be sure to go through your level again!

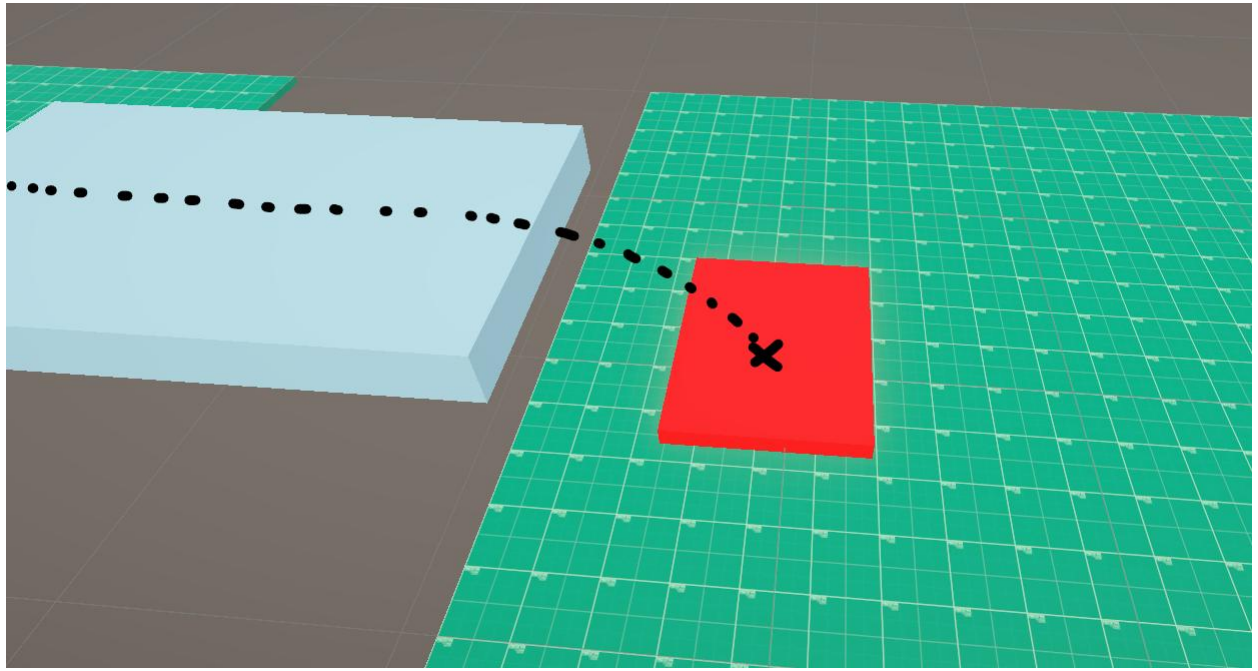
PRO TIP: Spamming jump is a great way to test this.



Design Bug: Telegraph All Damage

Having the player jump and not be able to see the kill box is bad design.

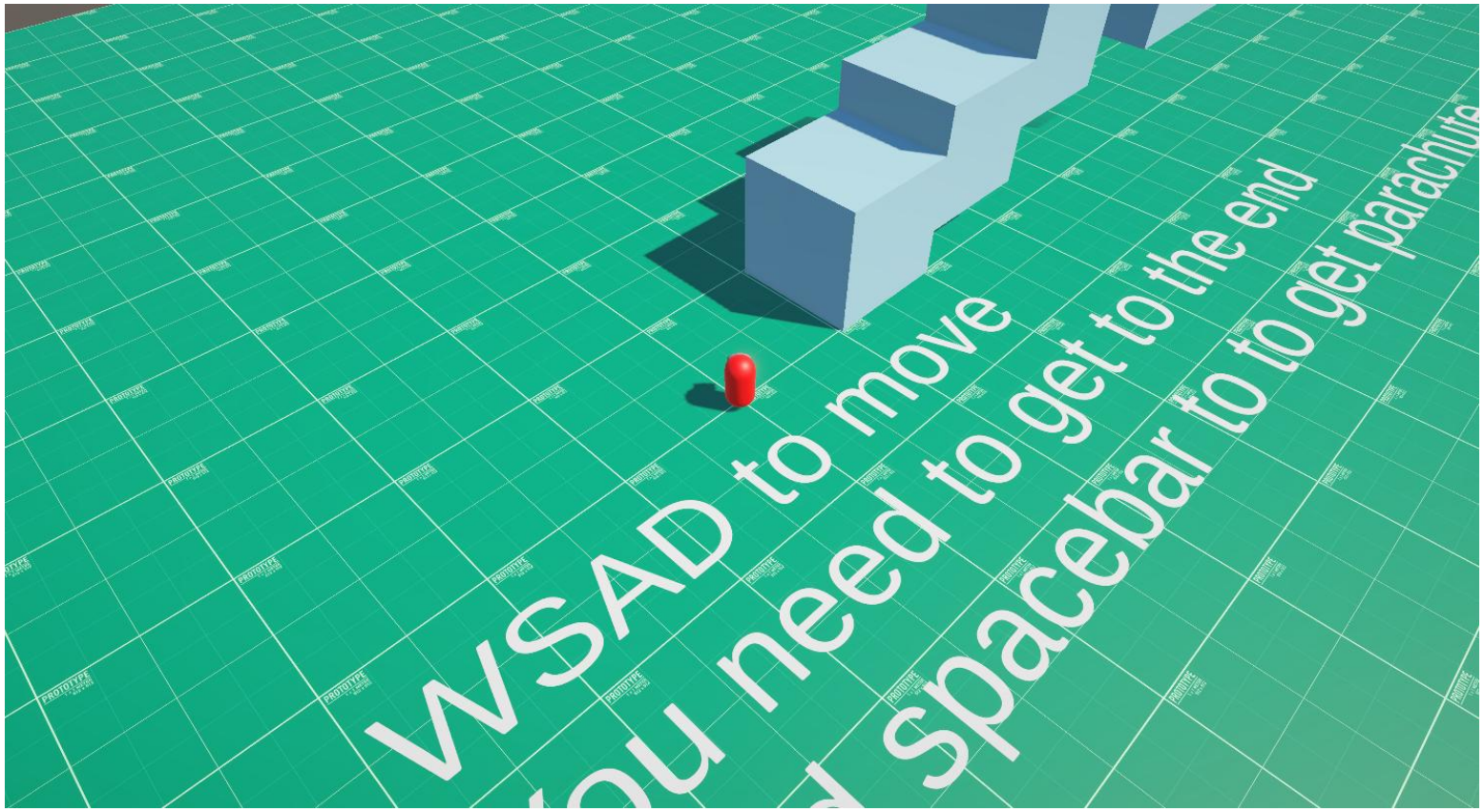
All damage should always be avoidable, given enough skill.



Accessibility Bug: Text is 45 degrees

This makes the text really hard to read properly. If the player is on a small screen, they will see even less of this text!

Make this a UI element so it is always readable.

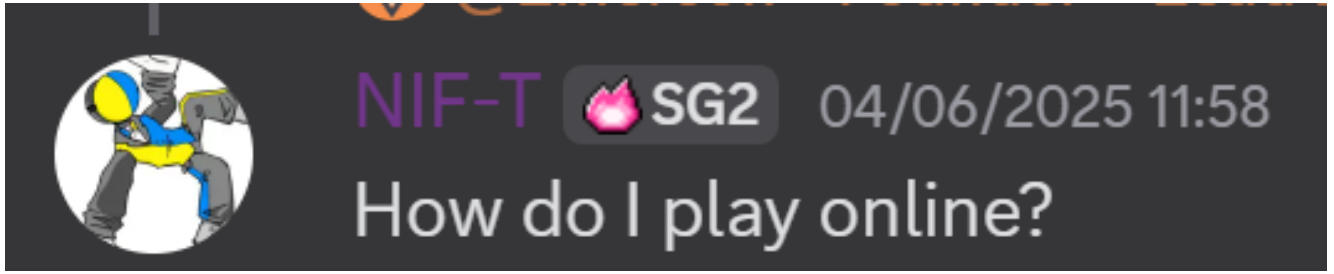


Accessibility Bug: High Performance

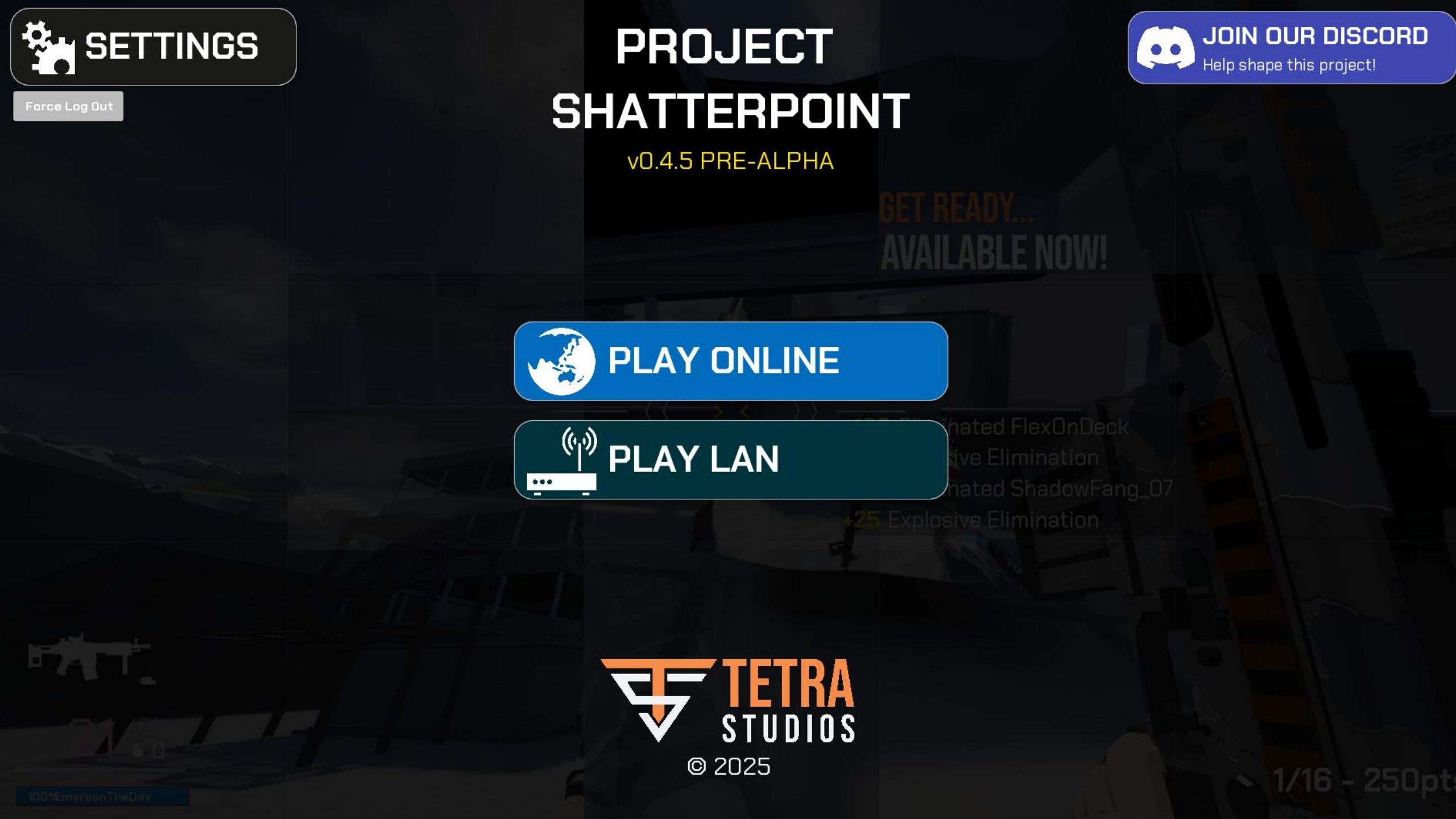
- The game runs, by default, on very high settings.
- Don’t get complacent and assume your audience will have the same hardware as you!

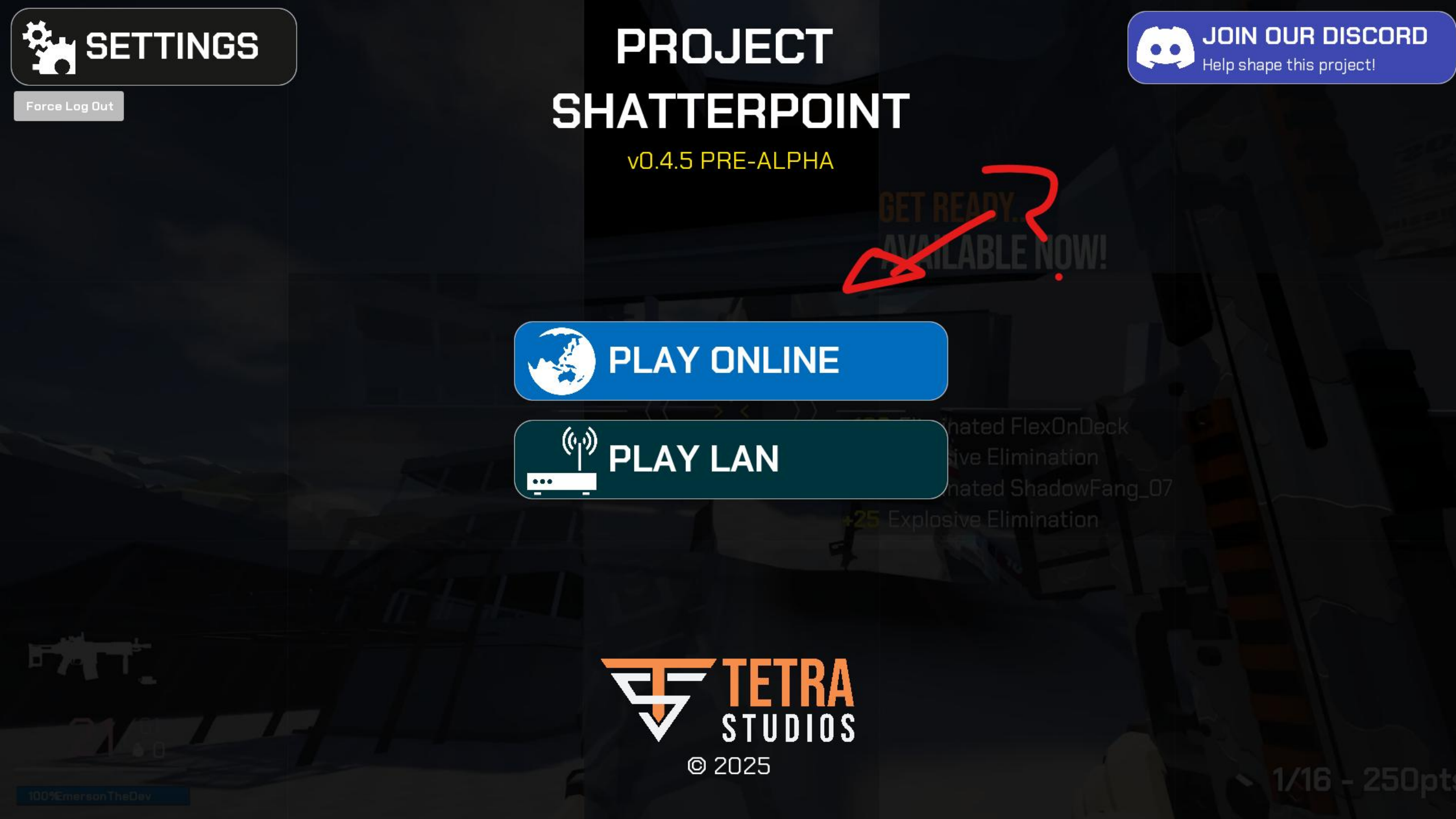
End of Part 2
Any Questions?

Part 3:
Reading Between The Lines

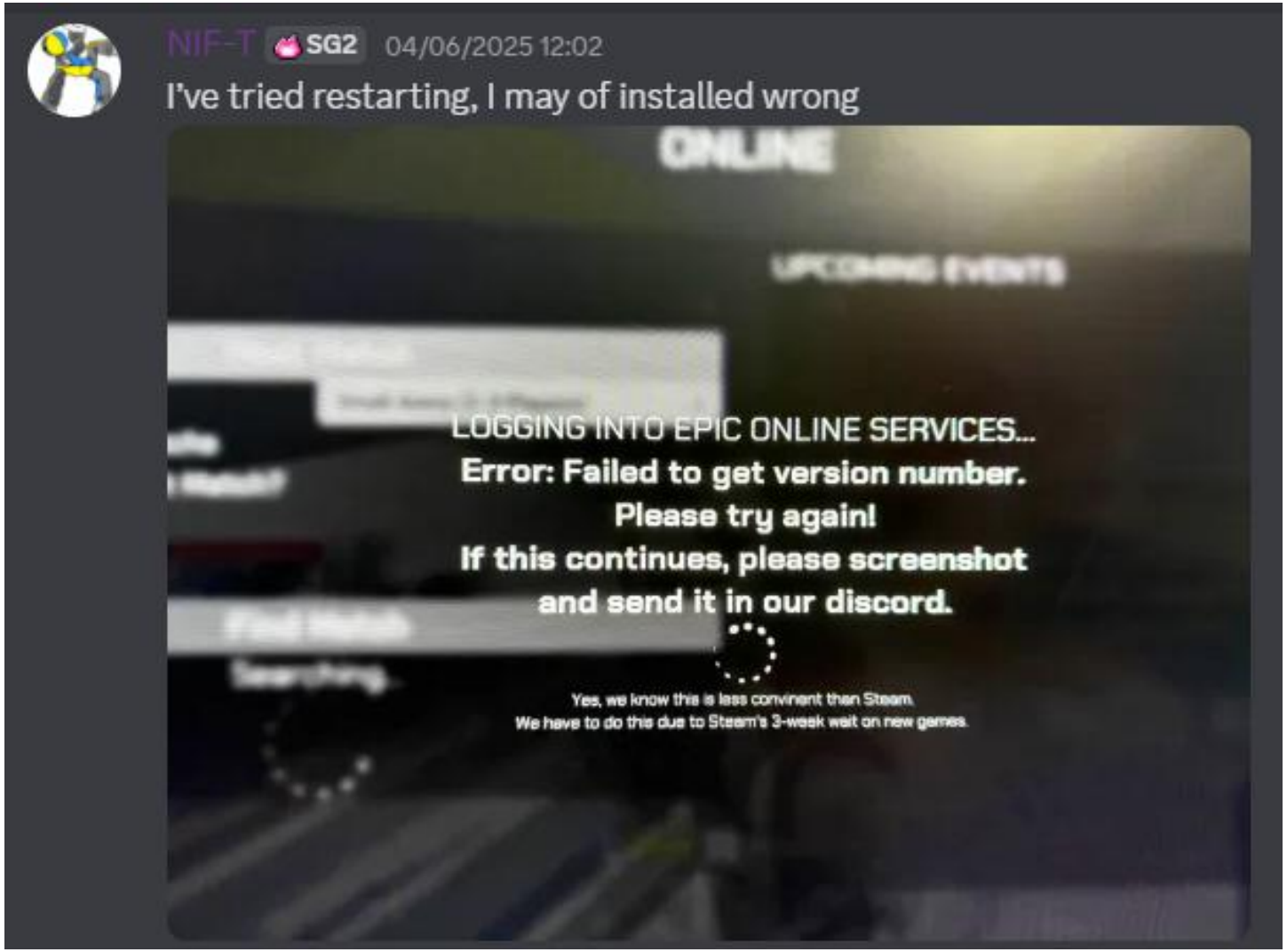


Is this a Technical or Design bug?


SETTINGS
Force Log Out
PROJECT
SHATTERPOINT
v0.4.5 PRE-ALPHA
JOIN OUR DISCORD
Help shape this project!
PLAY ONLINE
PLAY LAN
TETRA
STUDIOS
© 2025


SETTINGS
Force Log Out
PROJECT
SHATTERPOINT
v0.4.5 PRE-ALPHA
JOIN OUR DISCORD
Help shape this project!
GET READY.
AVAILABLE NOW!
PLAY ONLINE
PLAY LAN
TETRA
STUDIOS
© 2025

Reading Between the Lines

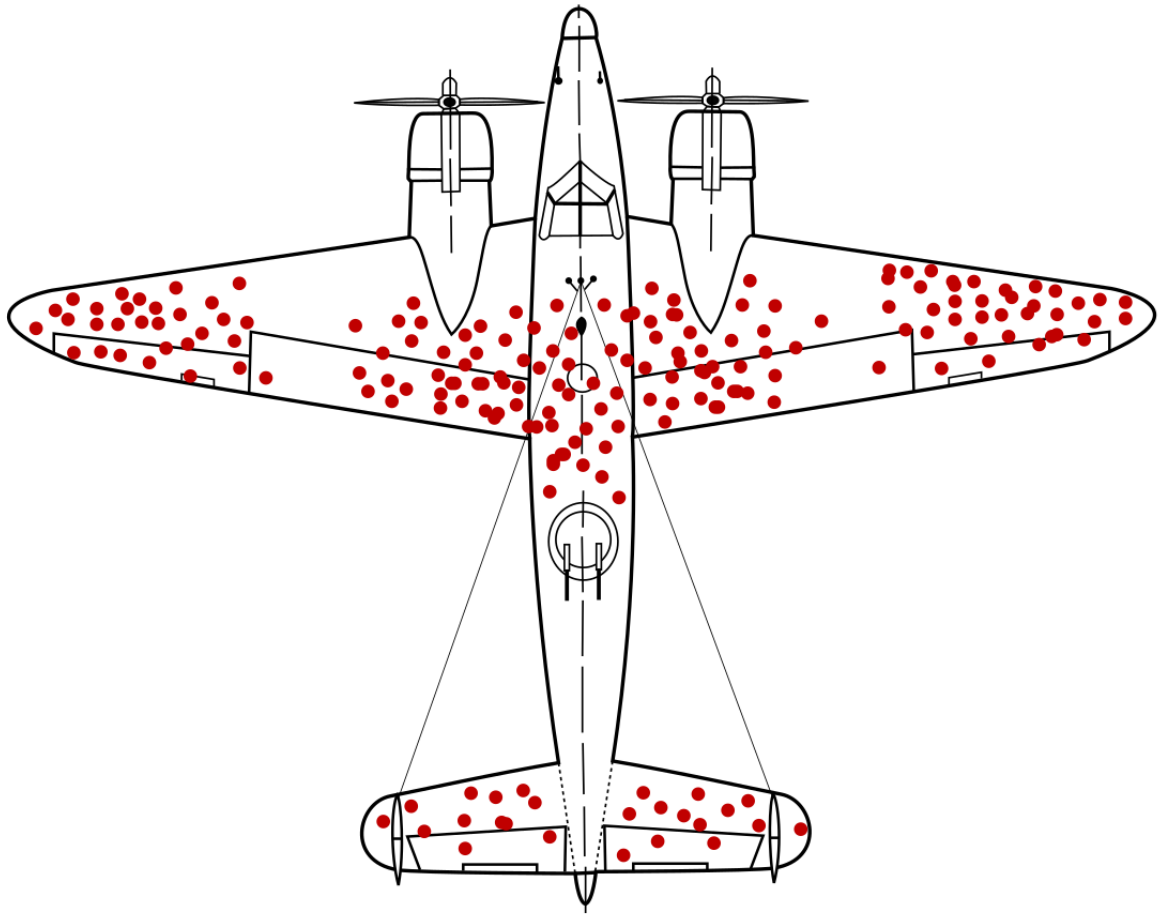


- This IS a Technical Bug, however the UX needs to be improved so the game can fail gracefully.

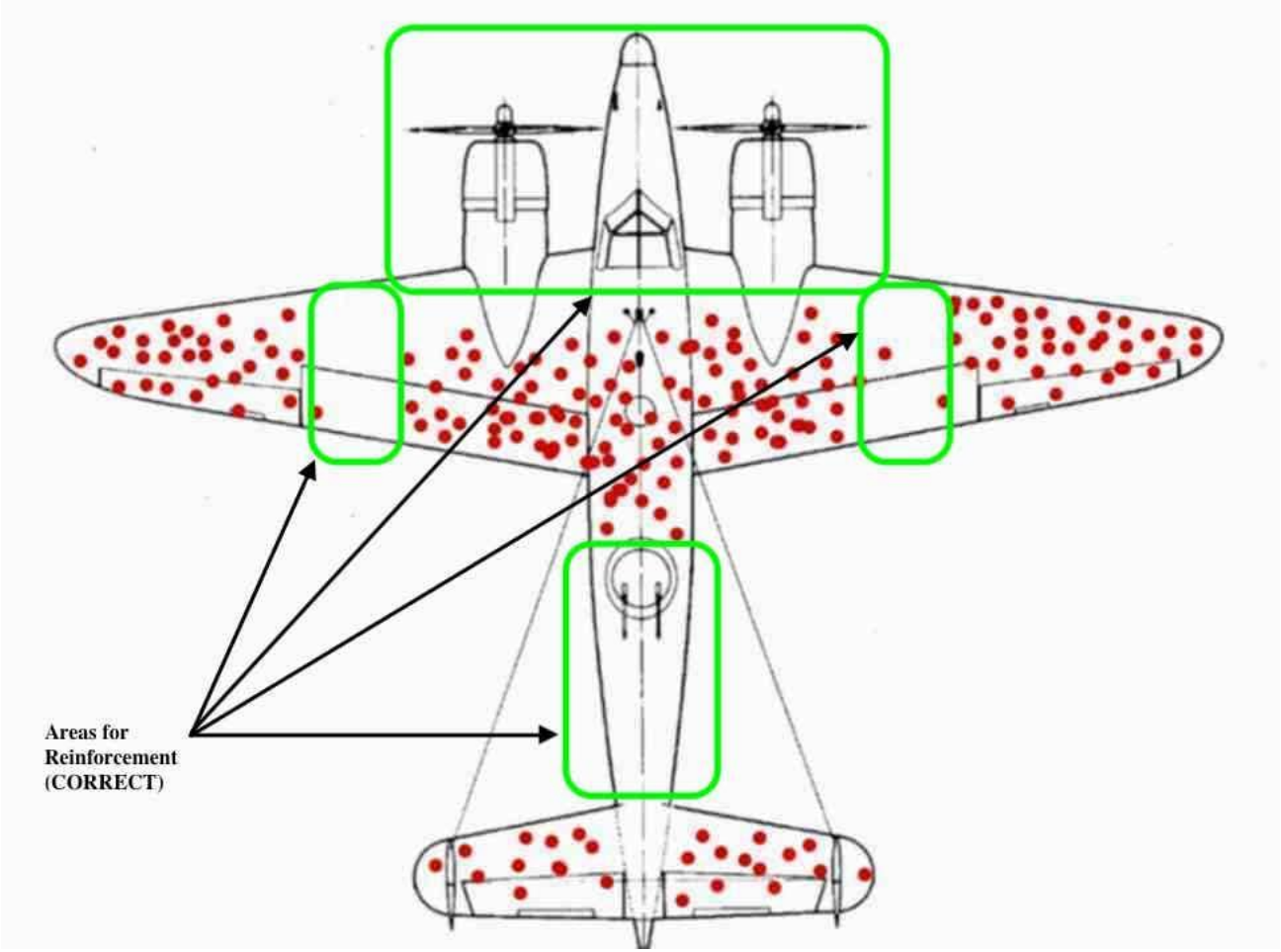
Reading Between The Lines

Players are almost always right about how they feel, and almost always wrong about their proposed solutions

Reading Between The Lines



Reading Between The Lines



Bugs can sometimes be good!

This streamer found a really big issue with the new destruction system – none of the buildings stuck to the floor!

However, the completely dynamic way the buildings moved in the level gave us a ton of ideas about new levels with fast-paced environment changes.

Edge Cases

Edge Cases - Latency

- You must always test with latency when developing a multiplayer game - have it on by default.
- Latency is tricky as it violates the fundamental understanding you've had for years – in multiplayer, any code may run later than you expect.

Edge Cases - Latency

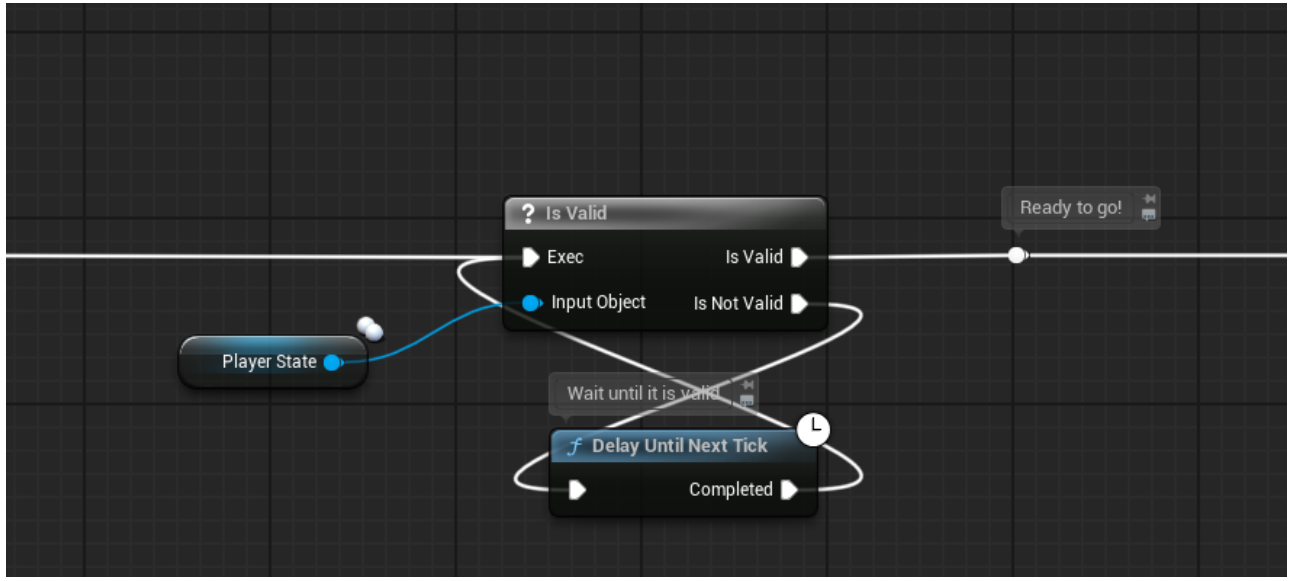
- Do not use delays!
  - Delaying a piece of code by 0.2s means it won't work for anyone over 200ms ping!
- Setup event-based architecture instead:
  - Server send message to client
  - Client performs action
  - Client RPC's to server telling it that the action is completed.
- Expect most players to be playing with <300ms ping.

Edge Cases – Joining

- When joining a match, ensure the player “waits” long enough until everything is loaded.
- EG: In Unreal, the load order is:
  - Hosting Player Controller connects first
  - Game Mode
  - Game State
  - Player State
  - Level
- BUT, this isn’t always consistent!
  - Other players can connect at any stage here
  - Order can sometimes differ between Editor and Packaged Builds
    - Sometimes the Game Mode is ready before the Player Controller!

Edge Cases – Joining

- So, running something like this means that you don't proceed until you know that the game is good to go.
  - Ideally this is event-driven, and not running on a tick delay, but this is a MUCH better solution than adding in a hard delay of 0.2s, for example.
  - The object you wait for could be an object in the level, which is at the end of the load order, so knowing this is valid means you know everything else is good to go.
- Moral of the story: Study execution order by using breakpoints and debug logs, and ensure your architecture can handle delays at any point during joining!



Edge Cases – Mid-Match Joining

- Setup your systems to allow mid-match joining FROM THE START!
  - Use replicated variables to hold values.
  - Once a client joins, take those values and “replay” actions
- Example: Building Destruction in Project Shatterpoint
  - Everytime destruction is applied, we save the data behind it to a SyncVar (Unity)/Replicated (Unreal) Array
  - Once a player joins, we fetch that data, and re-run all the destruction events using that data

Edge Cases – Fragile States

- The game is usually most fragile when you change states:
  - Joining a match
  - Possessing a character
  - Unpossessing a character
  - Swapping levels
- Apply your previous learnings and go HARD when the game is in these states to find issues.

Edge Cases – Fragile States

- This is especially true for “Seamless Travel”
  - Seamless Travel – moving players to a temporary “loading level” so they are not disconnected when swapping levels.
    - Seamless Travel begins, small temporary level is created
    - Player loads into temporary level
    - Previous map is destroyed, new level is loaded
    - Once new level is loaded, transfer player to that level and destroy temporary level
- EG: Equipping a weapon when in Seamless Travel’s temporary level caused a crash, since the weapon reference was not cleared from when the previous level was destroyed.

Edge Cases – Frame Rate

- Ensure you test the game works at any frame rate!
  - Unity - *"application.targetFrameRate = X"*
  - Unreal – Execute Console Command *"t.MaxFPS X"*
- Common Issues:
  - Differences in movement speed:
    - Multiply movement with
      - `time.DeltaTime` (Unity)
      - `GetWorldDeltaSeconds` (Unreal)
    - Run movement on a separate loop
      - `FixedUpdate()` (Unity)
      - `Async Physics Tick` (Unreal)

Edge Cases – Frame Rate

- Wait, why run movement on a Fixed Tick Loop?
  - Floating point errors can easily happen when your FPS is too high.
    - Floats can only have so many decimal places before it stops working.
  - Example: In Project Shatterpoint, the grapple would stop working once you hit 200fps
    - The difference is movement per frame was so small that the game thought you weren't moving, and it cancelled the grapple
  - Ensures consistency between all players on all machines.

Edge Cases – “It works on my machine!”

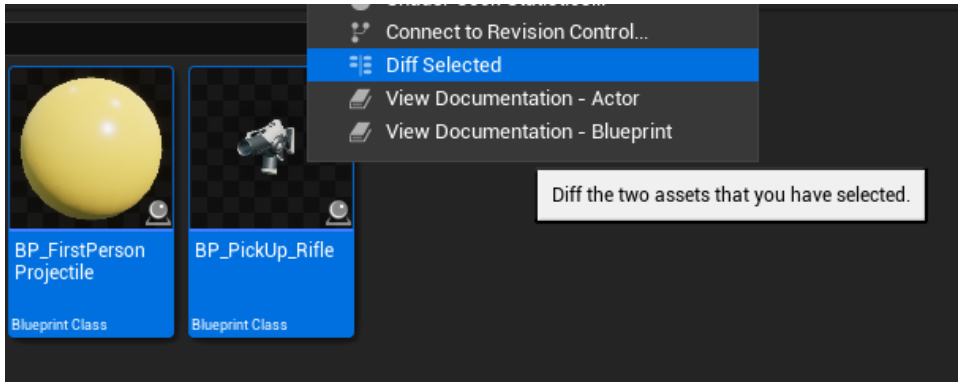
- Certain installation files are missing
  - EG Visual Studio components required for building
  - Android Studio SDKs not being on the correct version
- Library/Temporary Files being corrupted
  - This can happen way more often than you think!
  - Delete these files and try again.
  - Reclone the whole repo if needed.
- Run a Full Rebuild/Clean Build
  - Unreal
    - In Visual Studio, right click on the solution and click “Rebuild”
    - Delete Intermediate, Binaries and Build folders
    - Tick “Full Rebuild” and “For Distribution” when packaging
  - Unity
    - Delete Library folder
    - On the build dropdown when packaging, select “Clean Build”

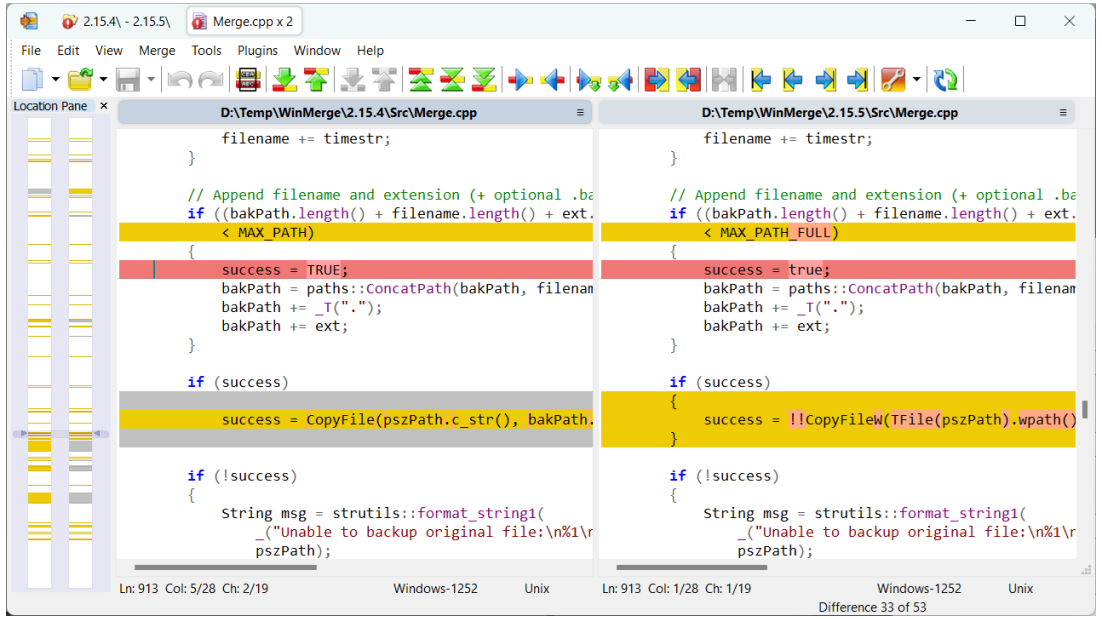
Edge Cases – Works on all but one

- This is when a certain issue only happens on once instance of the object (ie all weapons work but one).
  - Almost every single time this has happened, it is due to something being wrong with *that particular object*.
  - Don’t get fooled with going down the rabbit hole!
  - Check everything about that object matches with the working objects.
  - It almost always is due to a single variable being missing!

Edge Cases – Works on all but one

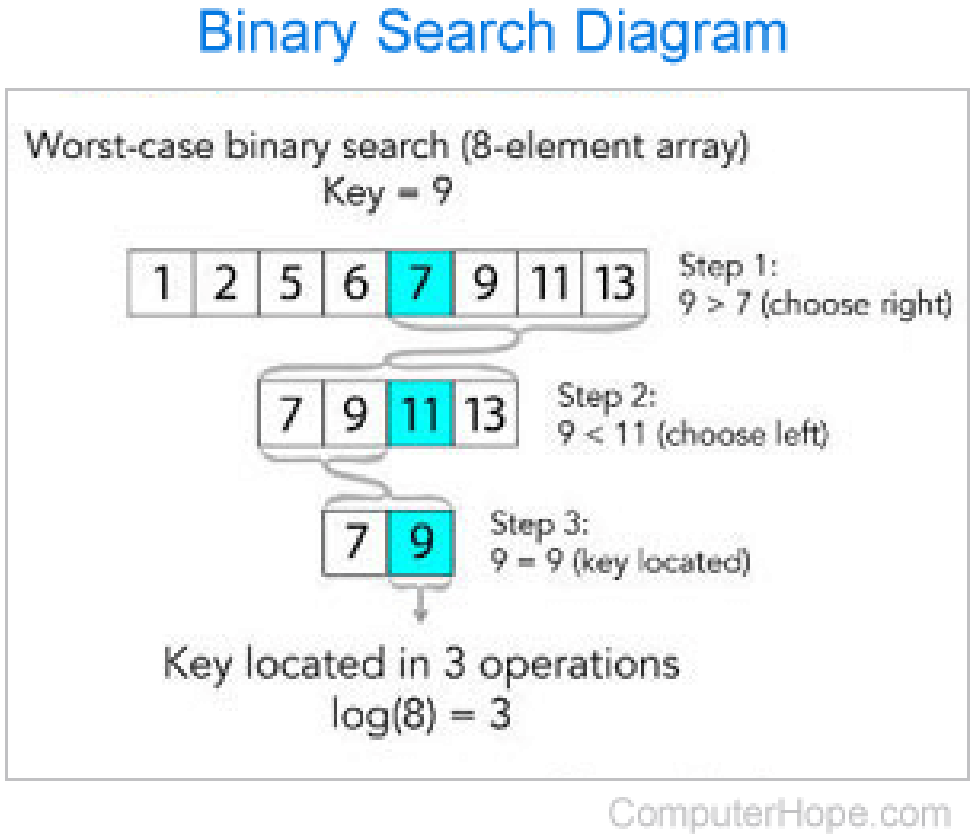
- Use:
  - Unreal – In-built diff tool
  - Unity – WinMerge
- To compare file changes between assets.





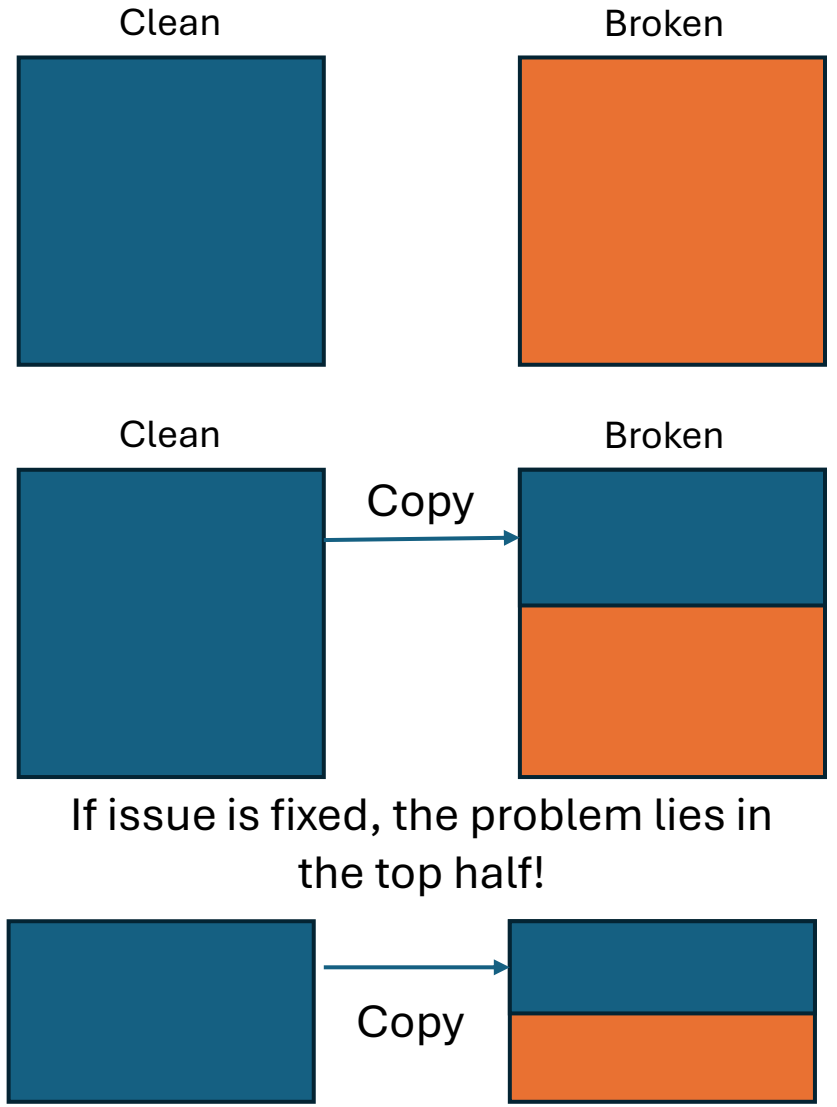
Edge Cases – Binary Search Changes

- If you have a working and non-working asset, use a binary search to find the problem issue.
- Binary Search
  - Split array into 2 halves
  - Is the middle value greater or less than the result you want?
    - Less: Get all left-most values
    - Greater: Get all right-most values
  - Repeat until you find the value!



Binary Search

- Likewise, start with the clean and broken asset.
- Copy the top half of the values from the clean and paste to the broken one.
- If the problem is fixed, you know the problem was in those top half of values. If not, the issue is in those bottom half of values.
- Split that half, and repeat until you find the problem line.



Edge Cases – Binary Search Changes

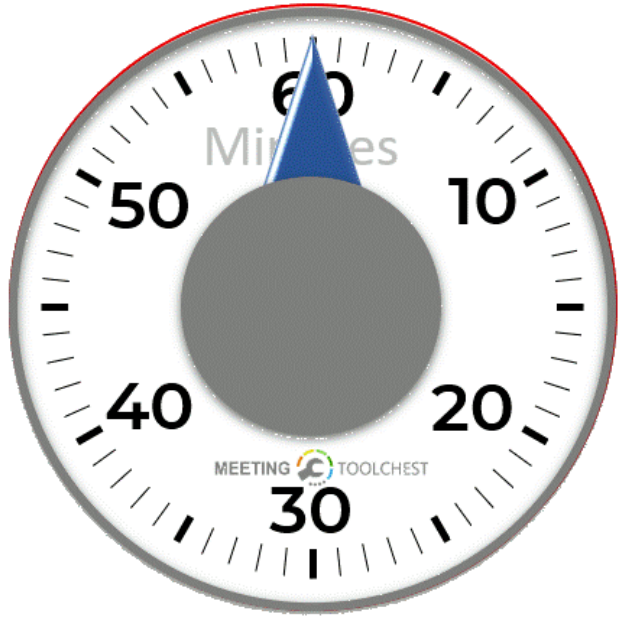
- Likewise, when fixing a bug, find the very start, and very end, and apply the same technique.
- EG Weapon firing causing null error?
  - Start at the place where you read input (mouse click)
  - And the end result (projectile spawned)
  - Now, disable the code halfway between these points.
  - Is the issue resolved?
    - Yes? Then the bottom half is the problem
    - No? Then the top half is the problem
  - Repeat!

Edge Cases – Never Assume Anything!

- Every assumption you make means you are likely skipping over the places where the bug actually happens.
- IE Weapon Not Firing Issue
  - If you assume the input is working properly and spend your time in the ammo systems, you entirely ignore the possibility that the input system is wrong.
- Do not assume any part of the system works until you verify each step first.
- Game Engines are great, but never 100% reliable.
  - **cough* Unreal *cough**

End of Part 3
Any Questions?

Part 4:
Putting it all together



60 mins

Try it yourself!

- Download this presentation and the project files
- Find and fix the bugs you come across – including Technical, Design and Accessible
- Feel free to change/add/remove anything you'd like to improve the game
- After the time is up, we will play each other's versions of this game.
- Ask me if you get stuck!