

GDV4000

Version Control with Unity



Introduction

- Problem-Based Learning
 - Identify the problem
- Adopting and utilising professional tools and practices to support student study.
 - Using industry-standard developer tools (GIT)
 - Practices (Using GIT)

Today's Problems:



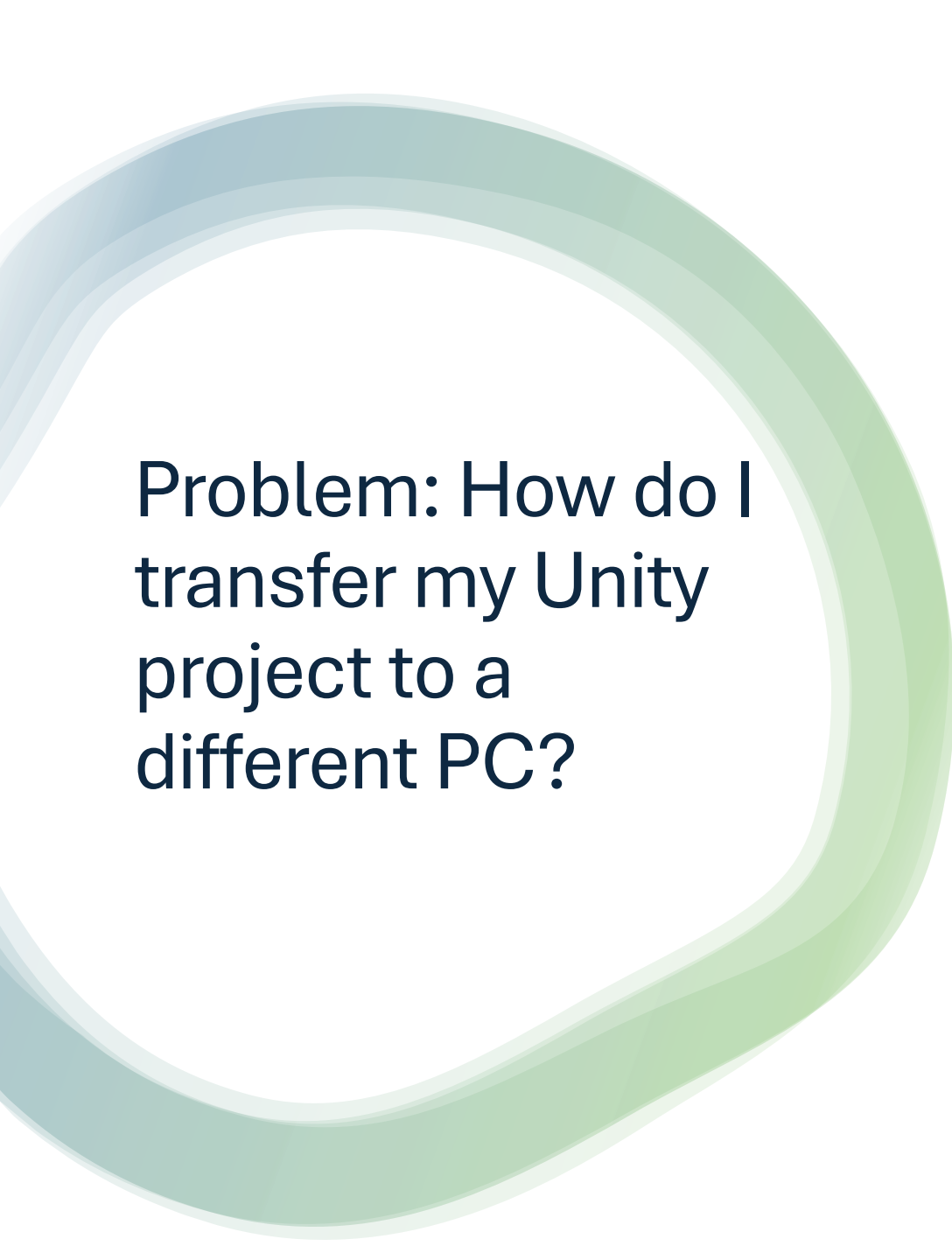
How do I transfer my Unity project to a different PC?



How do I share my project with my lecturer, for help with bug-fixing and assessment?



How can I roll-back to an earlier version of the project if it breaks?



Problem: How do I transfer my Unity project to a different PC?

- **Motivation:**
 - You may not have access to the same PC consistently in the lab, or may need to work off-campus (home/halls, whilst travelling etc.)
- **Solutions:**
 - USB Storage – Compressed project folder.
 - Danger – Loss of USB drive, potential corruption of drive.
 - OneDrive/Google Drive
 - Danger – Not designed for code. Large file sizes, could take a long time to upload and download. Not a suitable solution for group projects.
 - A Version Control System (VCS)
 - Used by software developers (including games developers)
 - Specifically designed to support development
 - Free/Education access

Version Control Systems (VCS)

Also known as

- Revision Control
- Source Control
- Source Code Management (SCM)

Used to manage multiple revisions of a single unit of information.

Revisions are typically in ascending order and associated with the person who made the change.

Can also be linked to a note or change log entry

VCS (cont.)

- Background
 - Developed from the 'Software Crisis' of the 60s, 70s, and 80s. Software was growing in complexity, but the procedures and practices were not evolving, leading to missed deadlines, increased workload, and budget overspend.
 - Borrowed from traditional engineering practices, such as blueprints and technical drawings.
 - Numbering system allows for tracking changes between versions.
 - Can return to an earlier version
 - Easy to identify the most current version




VCS (Cont.)

Numbering

- Numbers lower than 1.0 are usually development versions
- Major releases marked with a new primary number; other numbers denote bug fixing.
 - For example, UE 5.4 was the fourth distinct version of UE5, with new features added. UE 5.4.4 was the fourth bug fixed version of UE 5.4.

Large Distributed Teams

- VCS is designed for:
 - More than one user
 - Teams that may be spread across different geographical locations.
- This can result in two or more people working on the same code at the same time, which can be a problem!



VCS (cont.)

- This can be solved by:
 - File Locking - Locking the file so it cannot be altered by anyone else
 - Version Merging. Merging new revisions into the existing file. There needs to be an agreement on who has the most complete and up-to-date version.
 - Distributed Version Control

Aside: A Brief History of Source Control Systems

Local Version Control

- Centralised (Hardware/OS – File storage)
 - Mainframe era.
- Developers' 'checkout' files – Locked to a single user.
- Used for individual projects
 - Versions of documents (coursework etc.)
 - Identified by version number or a timestamp

• Advantages

- No complicated merging of multiple changes to a file

• Disadvantages

- Low productivity
 - Only one developer per file
 - What about absence/
 - sickness?

Aside: A Brief History of Source Control Systems

Centralised Version Control

- Decentralised (Hardware/ multiple OS's)
 - PC on network/internet.
- Developers' 'checkout' a local copy
 - Changes made to local version
 - Changes merged back to central version
 - Partly automated – can require intervention.
- Each merge is a separate update
 - Can be tracked

• Advantages

- Developers can work on files in parallel

• Disadvantages

- Low productivity
 - Central point of failure
 - Access to server required (limits capacity to work off-site)

Aside: A Brief History of Source Control Systems

Distributed Version Control

- No need to check-in/ check-out
 - Each dev has a complete repo (copy of all the files)
- Changes are distributed by merging repos
 - Each dev merges their changes to the source branch
 - Can be multiple sources
 - Can combine developer branches independently

Advantages

- Developers can work on entire projects in parallel
- Can work on an individual repo (and track changes) without access to the server
- Popular for open-source projects, such as Linux kernel

Disadvantages

- May encounter file-size limits and pay-walls.

GIT

What is GIT?

- Distributed Version Control System
- Developed by Linus Torvalds

Why GIT?

- It's fast
- Fully Distributed
- Designed to work with huge projects

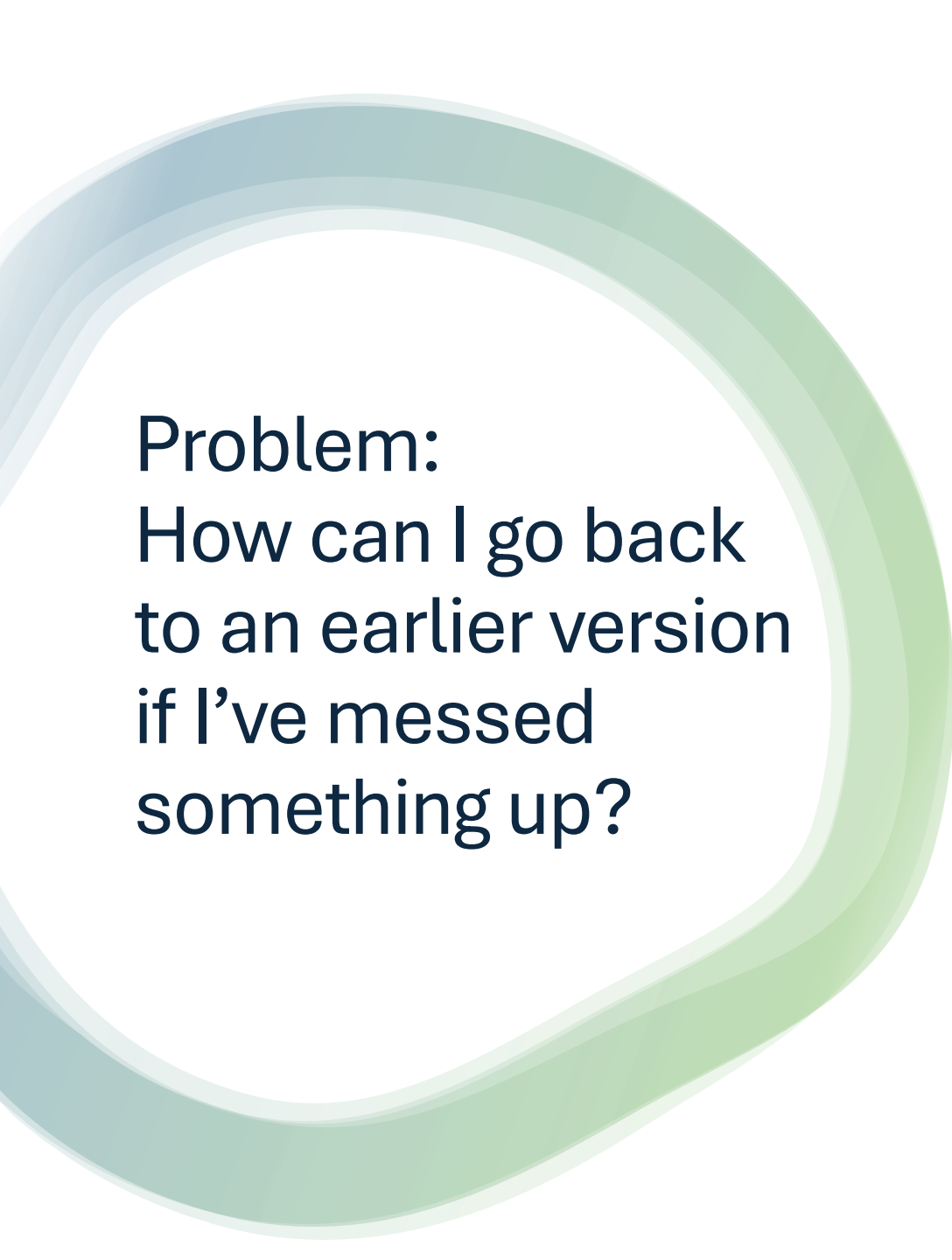
Popularity

- Well-documented with plenty of support available online
- Free cloud-based repos available.



Problem: How do I use GIT?

- **Motivation:**
 - We want to use an industry-standard solution to address the problem of storing and transporting our work – We have decided to use GIT.
- **Solutions:**
 - GIT can be command line based (does also have a GUI version, but we'll start with the command line)
 - Used on 'headless' (no GUI) servers
 - Continuous integration/ build release etc
 - Worksheet and Demos to help as a guide. And most important – **Practice!**



Problem:
How can I go back
to an earlier version
if I've messed
something up?

- **Motivation:**

- There are instances where you'll want to return to an earlier version
 - You may want to track the changes on a single file
 - The whole thing is broken beyond belief, and we need to go back to the last working version!

- **Solutions:**

- GIT
 - Demo
 - Worksheet
 - **PRACTICE!!!**

Summary...

- GIT
 - **Essential** tool – You will be using this all the time over the next three years!
 - Requires practice. Consider this a new skill to master.
 - Limitations
 - Designed for code (text)
 - Works for Unity, *largely* works for Unreal (there are plugins to make it easier)
 - Do not merge versions!
- Next Week...
 - First game in Unity.