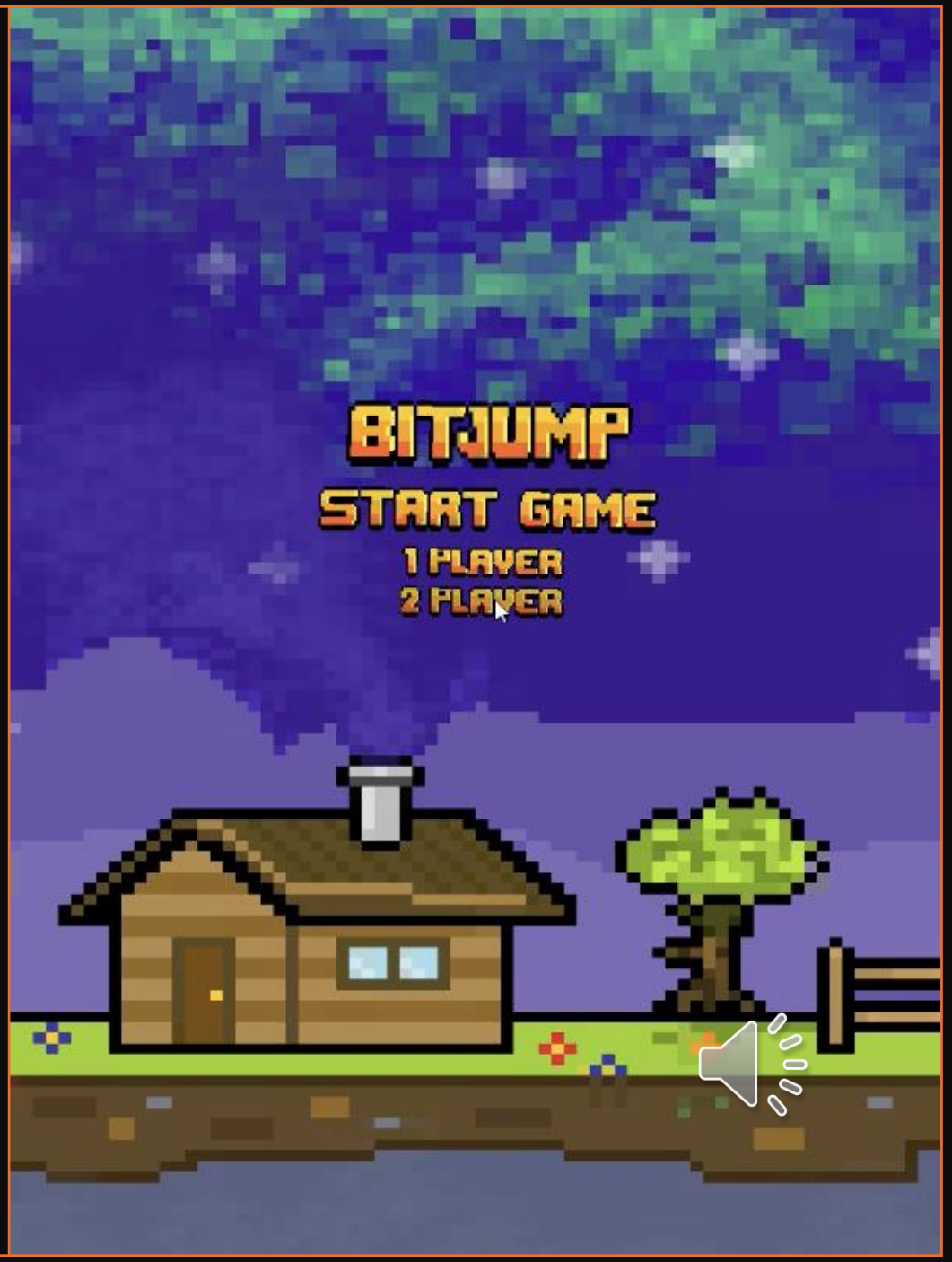
BITJUMP

Multiplayer Networked Game



Content

- Aims and objectives
- Background and Context
- Analysis of Game Genre Popularity
- Comparison of Networking Framework
- Preparing Unity for Photon Networking
- Photon Applications in Unity Games
- Video Demonstration
- Scene Management, Synchronization and innovation
- Loading Screens and Testing
- Results and Discussion

Introduction: Aims and Objectives

01

In-depth analysis of previous literature regarding networking frameworks and their integration into game-engines.

02

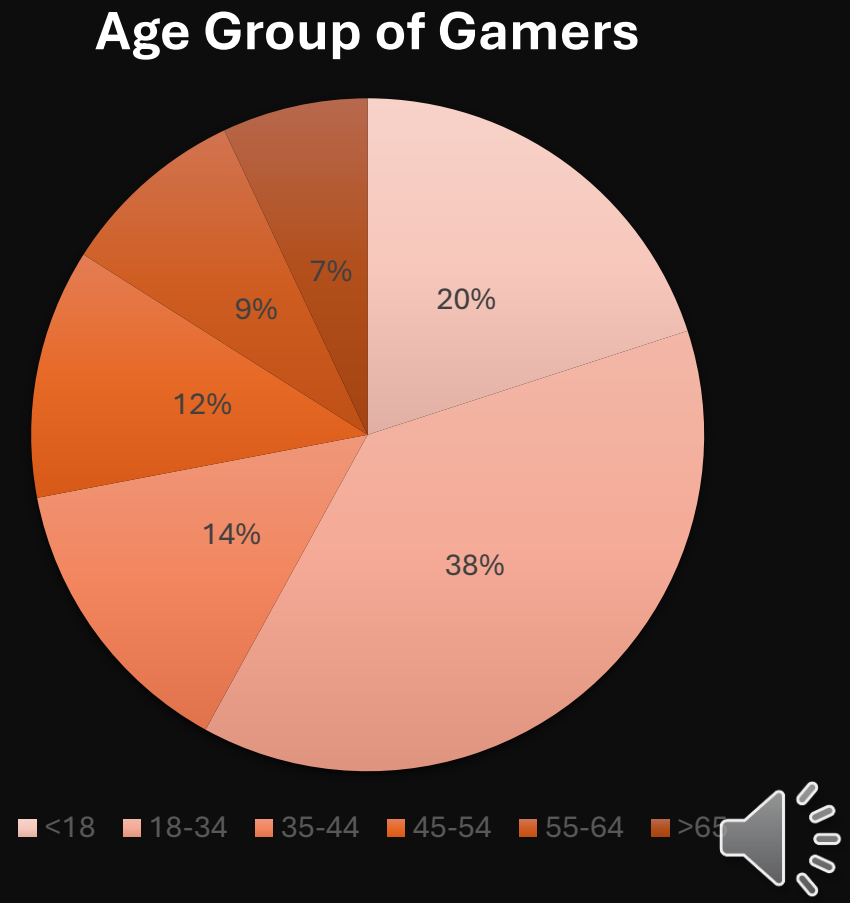
Further research into optimal game mechanics, and how multiplayer can be integrated successfully.

03

Develop a seamless multiplayer title, providing users enjoyability and contributing my own technical analysis to the field of game-dev.

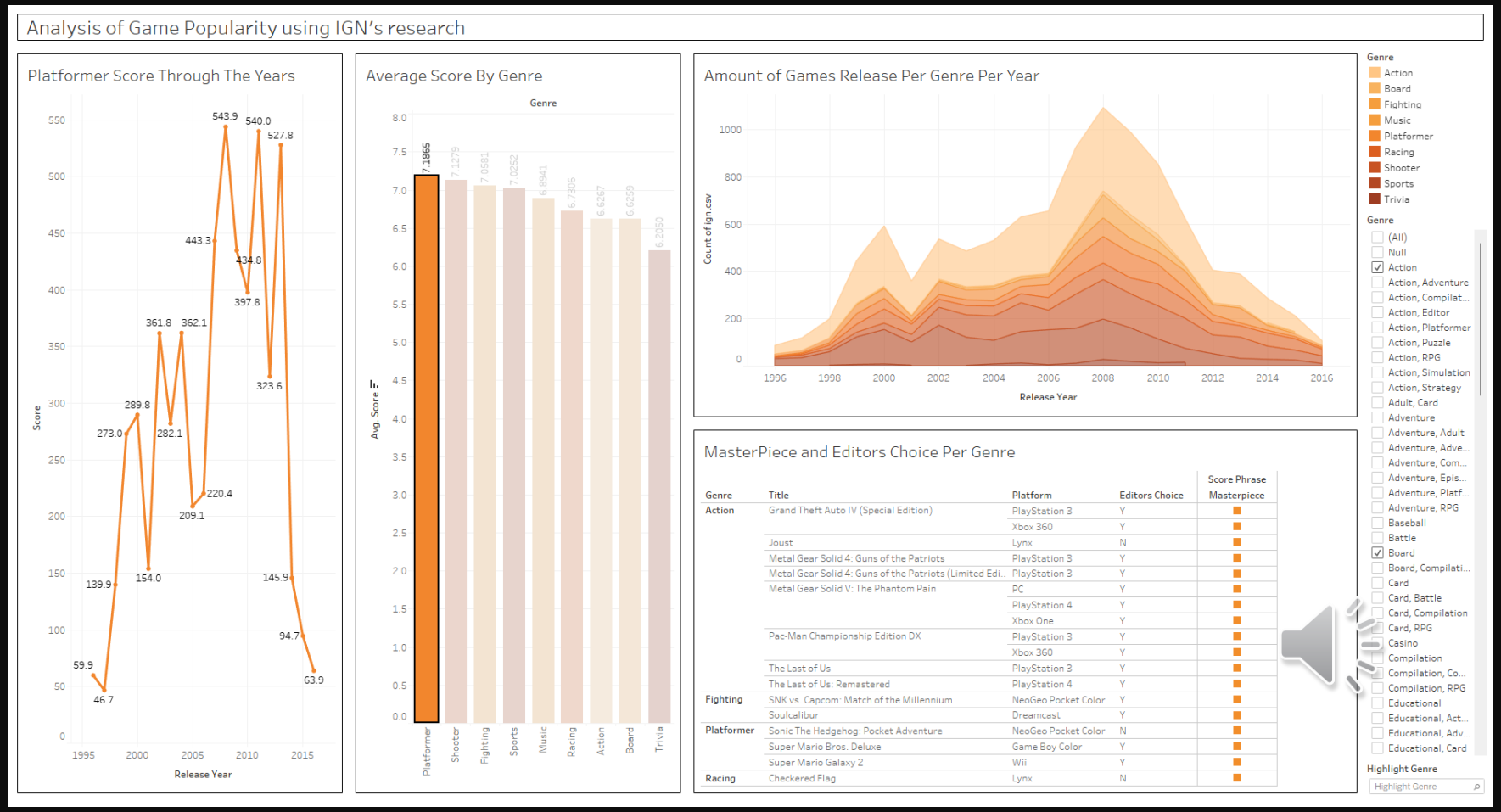
Analysis: Background and context

- My inspirations sparked off both primary and secondary research means.
- The demographics reveal that platformer games were popular, with the corresponding figure suggesting most gamers being born during the genre's surge in popularity.
- The ideology of gaming has always been a dynamic force, consistently evolving, with further research on new innovative strategies, which I wanted to contribute to.



Analysis of Game Genre Popularity

- The corresponding tableau workflow portraits how the genre of platformers have a consistent reign of popularity, resonating with the nostalgic feel for older players.
- With 52% of gamers being over 18, all would've experienced the surge in platformer popularity beginning in the early 2000's.
- IGN's dataset also portraits the Platformer genre to have a high proportion of 'Masterpiece' titles gaining IGN's editor's choice.



Comparison of Networking Framework

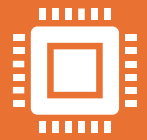
The table below indicates that Photon and Mirror are optimal frameworks, offering advantages over competitors. This report will specifically focus on Photon due to the simplicity of integration into a Unity environment.

Feature	Photon PUN	Mirror	UNET	FishNet
Documentation	Extensive	Extensive	Extensive	Limited
Ease of Use	Beginner-friendly	Beginner-friendly	Beginner-friendly	Intermediate
Network Architecture	Client-server	Client-server	Client-server	Client-server
Protocol	UDP	UDP	UDP	UDP
Latency Handling	Good	Good	Good	Good
Reliability	High	High	High	Medium
Scalability	High	High	Medium	Medium
Community Support	Strong	Strong	Strong	Limited
Price	Freemium model with paid plans	Open-source with paid support plans	Server hosting $5 a day dependant on CCU.	Free and open source

Preparing Unity for Photon Networking

- For Unity to work with Photon, the Photon PUN 2 package will need to be imported from the Unity Assets Store.
- Photon will then require slight configuration, including the setup of a new App ID on their website
- This would then allow for the Photon. PUN library to be used within scripts, which opens to an array of different networking functions.

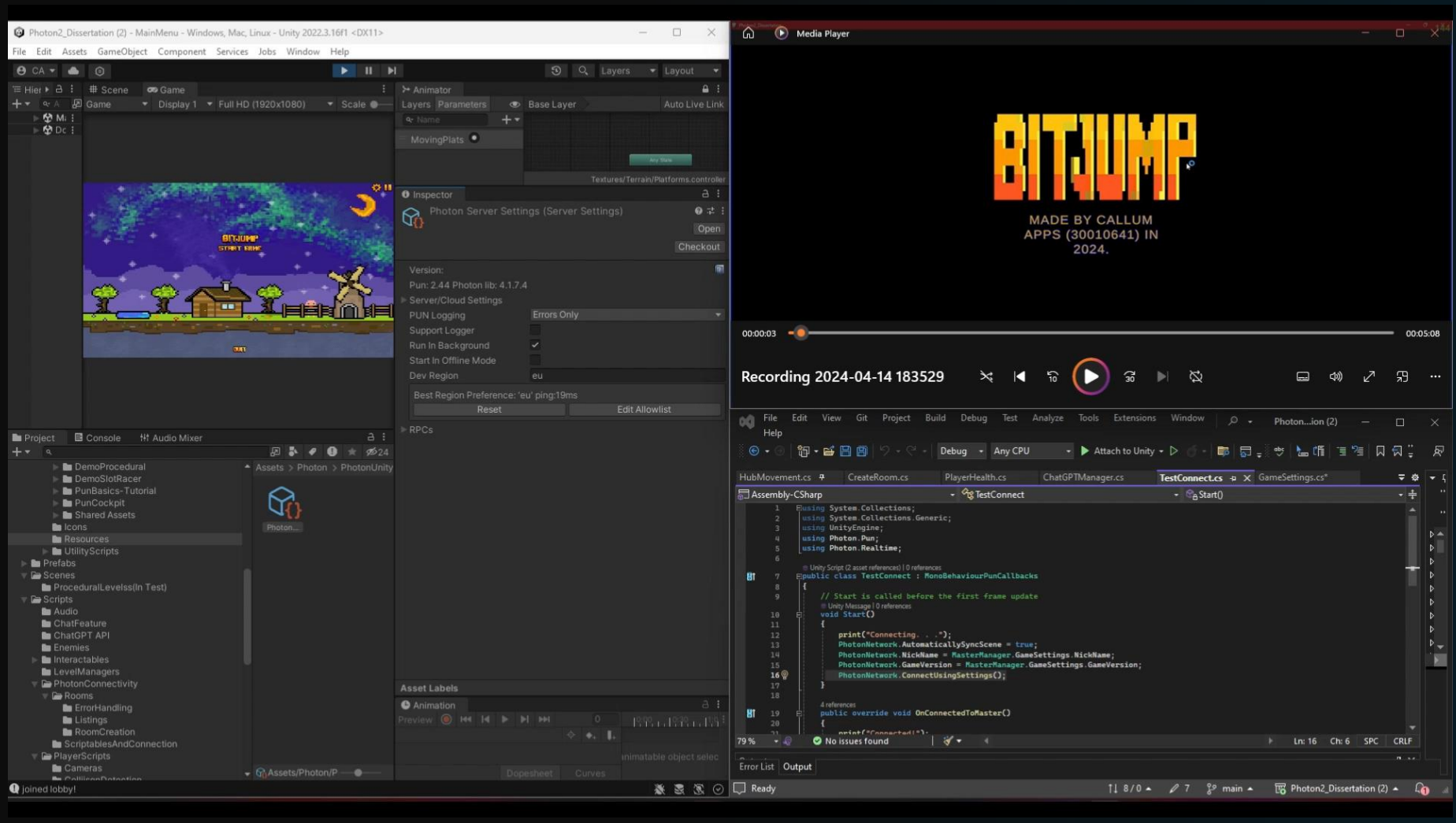
Methodology: Photon Application in Unity Games



Within BitJump, Photon was responsible for the creation of Nicknames unique to each user, connecting users between servers, and synchronizing player movement and animations.

This was achieved through a variety of Photon components, being PhotonView, PhotonAnimatorView, PhotonTransformView and RPC calls.

Video Demonstration



Scene Management, Synchronization and innovation

To ensure seamless multiplayer gameplay, the Photon component: Photon.LoadLevel() and Photon.AutomaticallySyncScene, had to be utilized.

This would ensure that the scenes are synced across the server to all clients, rather than just one client transitioning to the next scene, both would transition.

BitJump contributes towards the ever-growing innovative studies of game-development using up to date methods and contributing towards the field.

Loading Screens and Testing

Loading Screens were implemented between scenes to ensure that each client has connected to the Photon Servers, this was required to handle the error of users not being connected prior to creation or joining of a game.

Error handling was also a requirement within my application, this would reduce the odds of the game crashing when a user does an action that would throw an error, instead a dismissible error message displays onto the players UI, e.g. ”Game is full” “Waiting for Host”.

This was also used to stop errors from being thrown when players who aren't the host try to launch the game.

Discussion and Results

After development and research was complete, results were composed of a fully functional multiplayer title (BitJump) incorporating innovative features such as seamless multiplayer and integration of the ChatGPT API.

In addition to this, further analysis were conducted via means of a webpage, questionaires and user testing, to evaluate user feedback, therefore refining the multiplayer experience for optimal enjoyment and engagement.

Previous literature was also reviewed, leading to an understanding of optimal networking frameworks and their application into a Unity environment.