PORTFOLIO

JOHN JAMES GUTIB

PINK IS THE NEW EVIL! – PC RERELEASE

DESIGN GOALS

IMPROVED GRAPHICS AND NEW POSTPROCESSING
OVERHAULED MOUSE AND KEYBOARD CONTROLS
CONTROLLER SUPPORT
NEW SOUNDTRACK

I continued playing around with game development skills, despite having exited the industry. My most recent learnings consisted of Physically Based Rendering principles, in particular PBR materials, HDR lighting, and filmic tonemapping. I wanted to try implementing these in a game, and Pink is The New Evil was the perfect candidate for overhaul. Despite running on a 5 year old version of Unity (Unity 5.5.0f3), it had all the necessary features for my needs. Along with various tweaks and improvements, and a new soundtrack composed for the game by my brother, Aaron James Gutib, I developed and released the definitive version of Pink is The New Evil.

DEVELOPED USING **UNITY**
RELEASED FOR **WINDOWS**

4 MONTHS DEVELOPMENT TIME

FEBRUARY 2021 – PRESENT

An overhaul of my seminal personal project, improved, polished, and rereleased for PC.

Pink is The New Evil!
https://jamsers.github.io/Pink-is-The-New-Evil

GitHub Repository
https://github.com/Jamsers/Pink-is-The-New-Evil

PHYSICALLY BASED RENDERING

HIGH END RENDERING FOR PC

Implementing PBR was a 3 step process. Unity already has native support for PBR materials in the form of its standard shader, so most of the work was converting existing materials and creating new texture maps. I used Substance Painter to create the texture maps – since the artstyle of Pink is The New Evil is very simplistic, it was mostly a matter of polygon masking and assigning appropriate materials to the proper sections of the 3D model.

Next was HDR rendering. Actually enabling this was as simple as switching Unity's configuration to deferred rendering, switching to linear color space, and turning on HDR. However since the game was originally designed for basic mobile rendering, lighting needed to be heavily tweaked to look right under the new color space. A reflection probe was enabled to ensure materials had proper lighting information and looked correct.

Postprocessing was the last part of the setup and brought the whole package together. ACES filmic tonemapping was used to map the final image into sRGB, ensuring rich color and contrast. Bloom was enabled to handle bright light clipping, and chroma film grain was enabled to manage color banding. The final result is a cohesive image that reacts powerfully to lighting.

A scalability option has been introduced to allow the game to run on lower end GPUs.

OVERHAULED CONTROLS

MOUSE AND KEYBOARD CONTROLS FOR PC, AND CONTROLLER SUPPORT

Pink is The New Evil was originally designed for mobile phones, and optimized for one handed gameplay. Controls, triggers for the special attacks, and auto attacking all had to be overhauled and reimplemented for mouse and keyboard controls, and Xbox One controller support on PC. Special attacks can now be directly triggered with buttons, rather than the tapping combo system on mobile. Movement is now also direct and the virtual joystick system has been removed. Attacking now has to be done manually, rather than automatically.

Menu navigation with a controller has also been implemented, which allows a player to use a controller exclusively when playing Pink is The New Evil.

The camera system was also improved to add additional weight and momentum to player movement, and player attacks.

AND MANY OTHERS

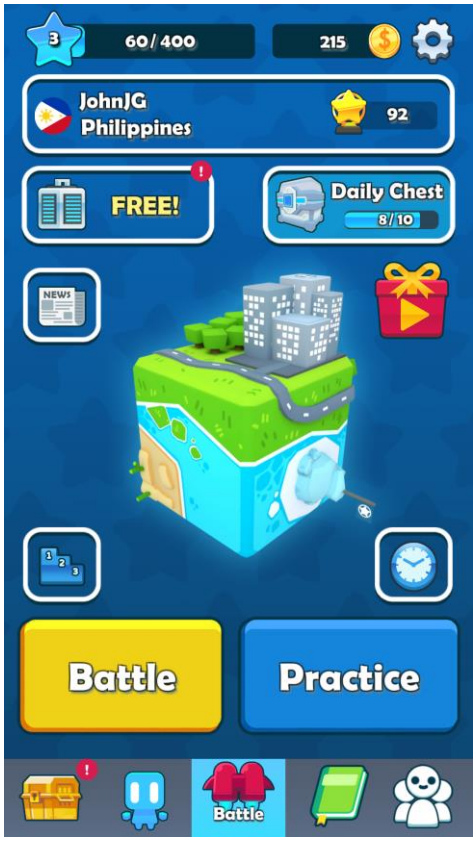
NEW SOUNDTRACK
VARIOUS MISCELLANEOUS TWEAKS AND IMPROVEMENTS
WINDOWS INSTALLER

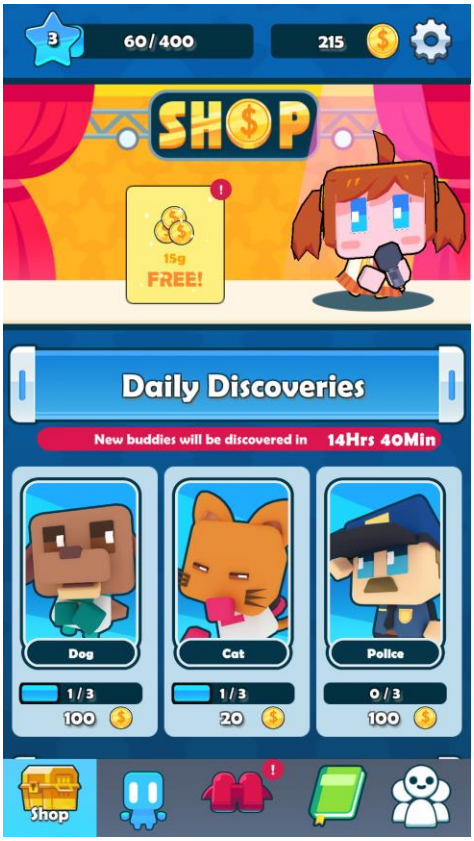
5 new tracks, tailored specifically for various sections of the game, were added in. New functionality was implemented to smoothly transition between tracks. Music was composed by my brother, Aaron James Gutib.

A litany of various bugfixes, improvements, and tweaks were implemented in the process of overhauling Pink is The New Evil, making the PC version the definitive version of the game. One such improvement is the addition of a toggle lighting button for survival mode, since PBR improves the nightmare lighting scenario significantly.

A proper Windows installer was implemented and deployed to ensure convenient installation and uninstallation from a user's system. A website for Pink is The New Evil has been set up, allowing for a centralized location for distribution and publishing.

WORD COSMOS





JUNIOR UNITY DEVELOPER

FEATURE IMPLEMENTATION
BUG FIXING
SERVER IMPLEMENTATION AND BUG FIXING

Word Cosmos was a big undertaking for FreCre, and development spanned 2 years. I was on boarded at the tail end of development, near the game's soft launch. I primarily implemented features designed by the spec team, fixed bugs both critical and non-critical, and as an incidental responsibility to certain features I've implemented and maintained, wrote Java code for a Google App Engine server.

DEVELOPED USING **UNITY**
RELEASED FOR **ANDROID** AND **IOS**

9 MONTHS EXPERIENCE

OCTOBER 2018 – JUNE 2019

My first experience as a professional game developer, using C# primarily.

Word Cosmos - Android
https://play.google.com/store/apps/details?id=com.frecre.wordcosmos

Word Cosmos - iOS
https://apps.apple.com/app/word-cosmos/id1406652331

PINK IS THE NEW EVIL!

DESIGN GOALS

ONE HANDED GAMEPLAY
FAST PLAYER MOVEMENT
DIVERSE ENEMY DESIGN

Right from the beginning, I knew the game structure would be heavily based on Call of Duty's venerable zombies mode, where you take on an infinite amount of enemies round after round. The final game ended up becoming its own unique idea – having 28 individually designed rounds that gradually introduce the different enemy types, weapons, and powers that the player uses throughout the game.

DEVELOPED USING **UNITY**
RELEASED FOR **ANDROID**

2 YEARS DEVELOPMENT TIME

AUGUST 2015 – MARCH 2017

My first complete project, created from the ground up all by myself – earning me valuable experience on game development.

Pink is The New Evil!
https://play.google.com/store/apps/details?id=com.john.project_one

GitHub Repository
https://github.com/Jamsers/Pink-is-The-New-Evil/tree/Android2016Release

GAMEPLAY

FAST PACED

Player movement is intentionally designed to be faster than every enemy in the game – allowing the player to easily outmaneuver enemy attacks and movements. Combined with fast attacks that don't interrupt or hinder player movement in any way and fairly high enemy damage, the player is encouraged to stay on the move, relying on hit and run attacks and good positioning to survive and defeat all enemy forces.

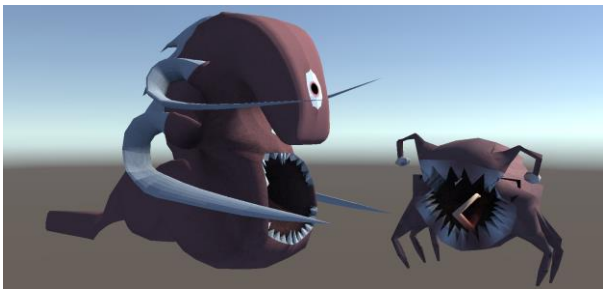
7 weapons are gradually unlocked through the course of the game, each one being better than the last. 2 powers are also introduced that add even more mobility to the player, making gameplay dynamic, with a sense of progression as the player tackles more and more levels.

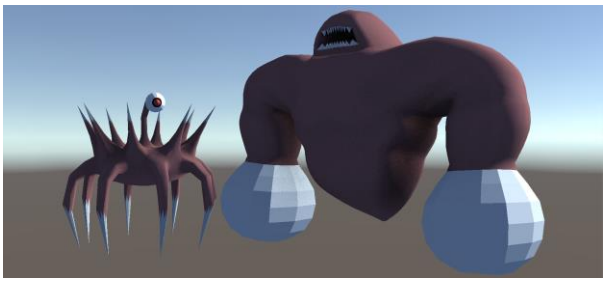
ENEMY DESIGN

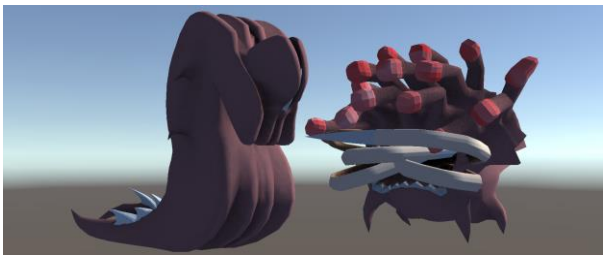
7 UNIQUE MONSTERS

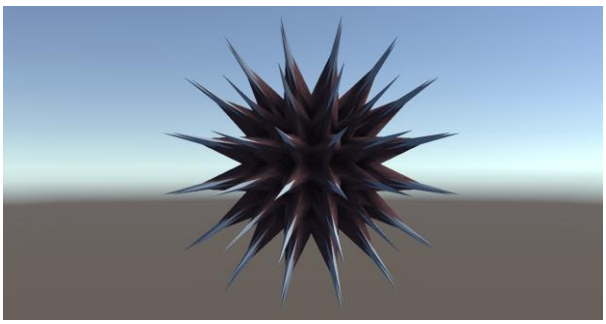
Each enemy is designed to have a distinct silhouette to make them easily recognizable from each other, and from the environment.

Every monster has their own unique way of attacking – the spikey ball, for example, simply rolls into you, while the big multi tailed snake creature will spit acid on the floor, providing an area of denial against the player. These unique characteristics keep the player on their toes for an exciting gameplay experience.









MAP DESIGN

ONE CONTINUOUS LEVEL

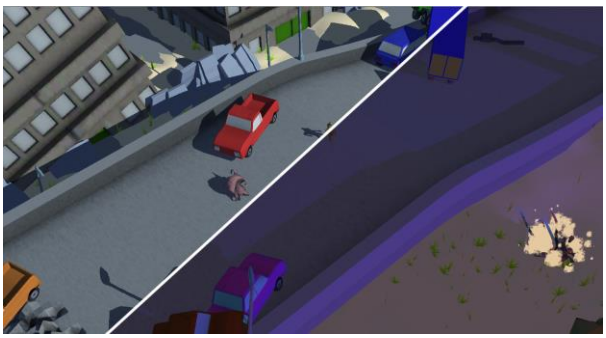
Instead of having gameplay be interrupted by loading screens or hitches every time the player progresses, the map is designed as one continuous level, with all the necessary changes to the level to reflect progress programmed as dynamic events that can be triggered when necessary. This results in a seamless experience playing through the game, resuming from a save, or unlocking barriers.

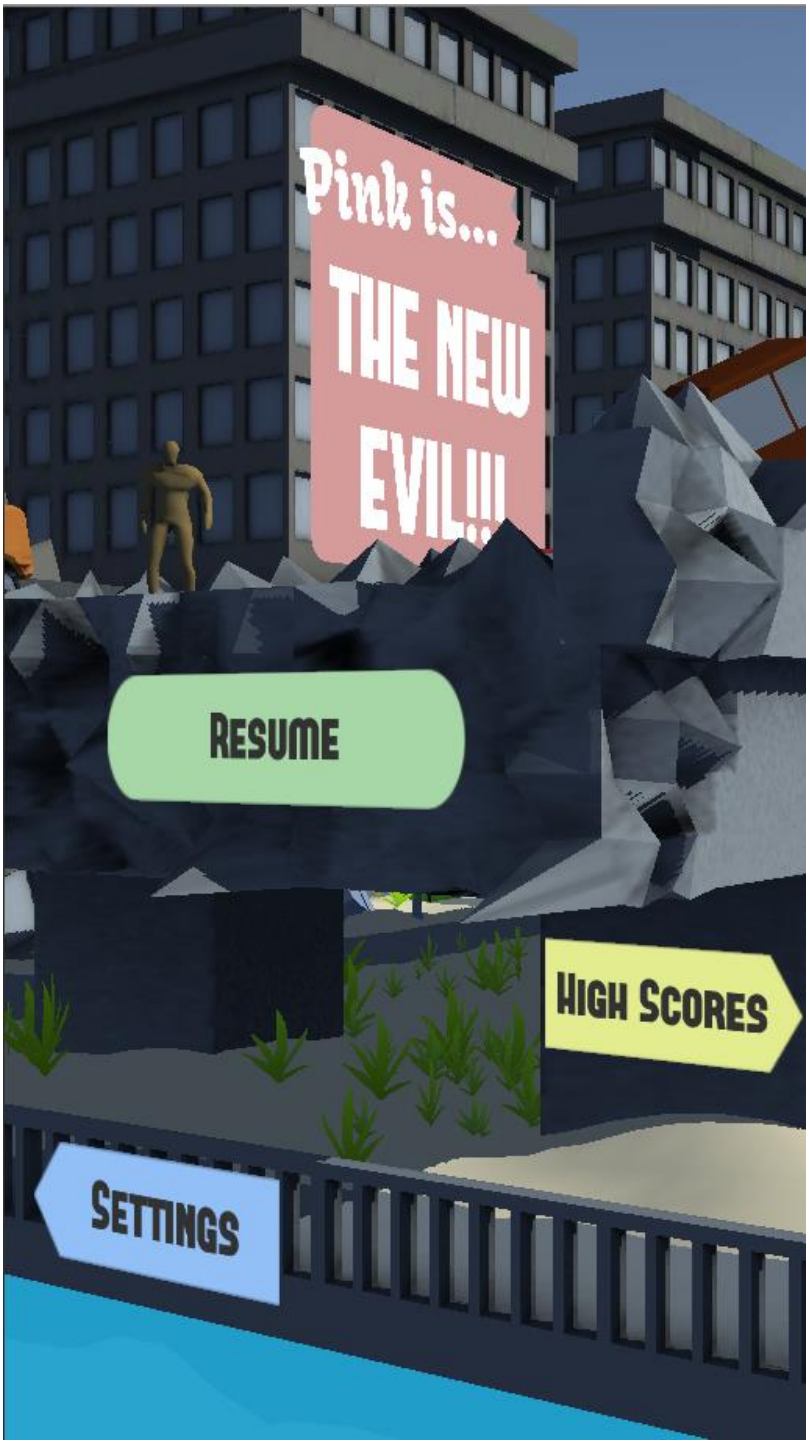
This also has the added benefit of allowing immersive progression though the game – now, whenever a barrier is lifted or a new weapon is unlocked, the game camera smoothly swoops over the point of interest, grounding the player within the world and explicitly showing the player where he can go next.

TIME OF DAY

VISUAL PROGRESS

Due to the decision to rely on dynamic lights instead of static baked lights, the lighting can now change in tandem with levels unlocked, resulting in a subtle, yet noticeable visual indicator of player progress.





PERSPECTIVE UI

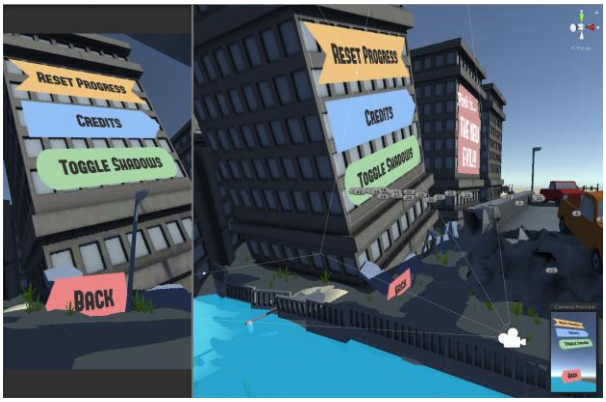
BOLD COLORS

Instead of creating a standard game menu, I decided to physically place the buttons within the game space, and use perspective and framing to create the menus within the game. This allows for a visually appealing design that uses simple shapes and colors for the buttons and UI elements, relying on camera angles and transitions to form the bulk of the menu work.

PHYSICAL MENUS

FUN NAVIGATION

The buttons of the main menu, along with all the sub menus they lead to, are physically placed in the game world, and navigating between them triggers the game camera to smoothly transition between pre-defined points. This makes for a dynamic, energetic look for the menu, and gives the game a unique, seamless feel, even before the player starts the game.



DESIGN GOALS

SIMPLE INFINITE RUNNER ON WATER

The game had to be simple and easy to make, but it needed to take advantage of mobile hardware as well, so that I could learn how to interface with the phone itself in order to take advantage of its unique inputs, such as the accelerometer and the touch screen. The game we settled on was an infinite runner in the same vein as Temple Run, but taking place on the ocean, which allowed us to skip having to design maps and randomization, opting instead to design a spawning system that could infinitely spawn obstacles and enemies. My brother, Christopher James Gutib, created the 3D models and textures for the game, while I handled the programming, design, and packaging for Google Play.

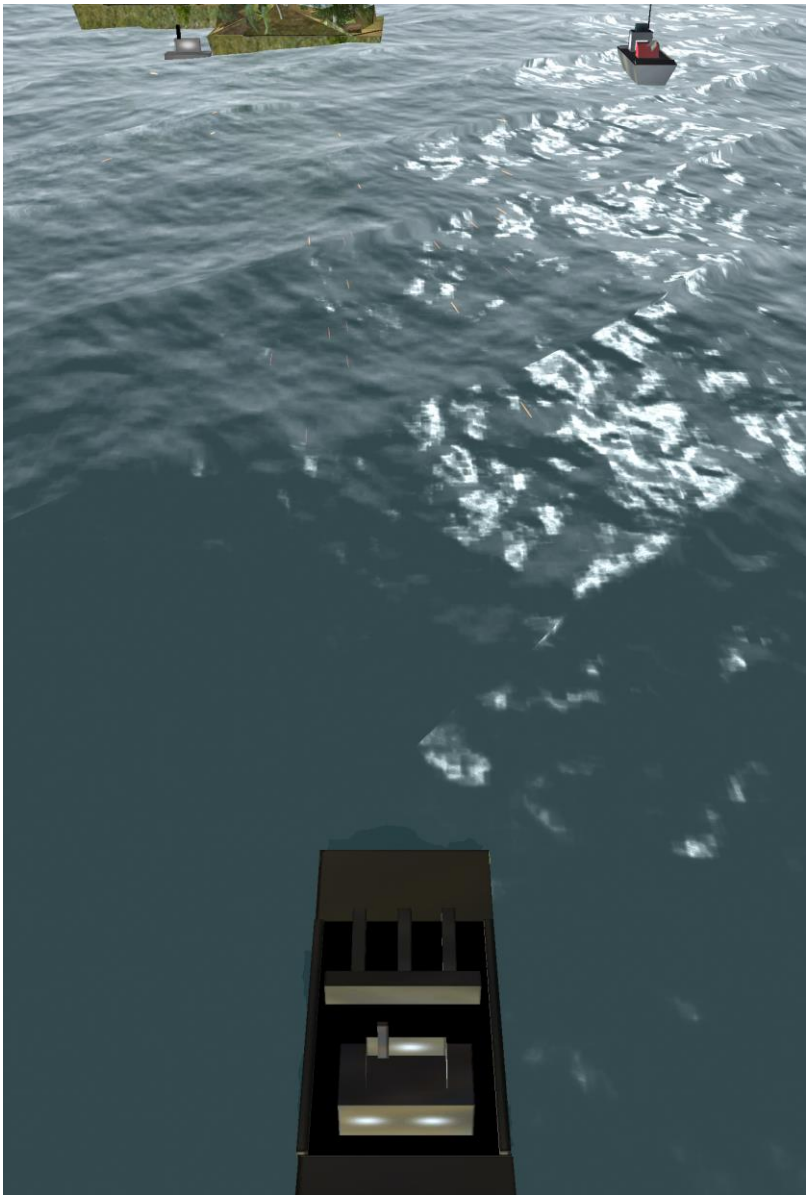
DEVELOPED USING **UNITY**
RELEASED FOR **ANDROID**

1 MONTH DEVELOPMENT TIME

JULY 2015 – AUGUST 2015

This was the culmination of the previous 2 years of self-study into video game development. An important personal landmark, it showed that I could create a complete game, and publish it to the world.

Battleships Clash!
https://play.google.com/store/apps/details?id=com.john.battleshipsclash



INFINITE WATER

HIGH QUALITY WATER ON MOBILE

Using Unity's built in high quality water, I scaled its feature set down to the bare minimum required to retain the undulating look to make it performant on mobile devices, while still looking like an actual ocean. A single, small size water plane was used, enough to cover the entire screen, and attached to the player ship.

This was enough to give the illusion of an infinite, deep ocean, giving us a wide canvas to create our spawning system, and giving the player an endless playground to dodge, destroy, and navigate in our game.

INFINITE ENEMIES

FILLING THE WORLD PROCEDURALLY

An area perpetually beyond the player's reach is defined as a spawning point, where enemies and obstacles are spawned at random intervals and positions. Each enemy has their own movement characteristics – boats will move faster than submarines, for example, while islands won't move at all.

All this results in gameplay that feels dynamic and unpredictable – the player is kept on their toes, dodging and weaving between enemies to survive. Yet, thanks to the infinite nature of the map, the player is never out of space to maneuver, allowing skill and alertness to take the player far, creating rewarding, challenging gameplay.