**CYBERBIKER (working title)**

**GDD (draft 001)**

*“The man-machine interface…at 200kph”*

**OVERVIEW**

* CYBERBIKER is an endless runner-style experience in which the player controls a cybernetically enhanced, motorcycle-riding warrior
* The player must navigate the biker through an ever-escalating series of reflex and spatial challenges whilst combating wheelbound enemies
* As the player progresses through the game, so too does the speed (and therefore the difficulty)

**NARRATIVE SET UP**

In the future, everyone **parks their brains in the cybergrid, while their bodies go to hell in the corporatized wasteland formerly known as Earth. In the grid, you can be anyone, do anything. Despite the utopian potential of this virtual world, there are those who want to see the whole thing collapse into a bundle of wasted ones and zeroes. These pro-corporeal anarchists are worse than computer viruses…harder to kill too.**

That’s where you come in.

As the CYBERBIKER, you have a license to delete. You don’t know who’s giving the orders to take the anarchists out, but you do know that their money is good.

So sharpen your blade, load your gun, and get on your bike.

It’s time to ride.

**GENRE**

Infinite runner (“infinite biker”)

**PERSPECTIVE**

3rd person, behind the back, centralized camera

**PLAYER COUNT**

Although CYBERBIKER is a single player experience, there is an asynchronous multiplayer component insofar as the player will compete for a high score on a centralized leaderboard system.

**TARGET PLATFORM**

Mobile

**DESIGN PILLARS**

Escalation

As a general rule, the game should get faster and harder as time goes on. This means that the player needs to play better to stay alive longer.

Pick Up And Play

As a mobile game, the need to get players in the game as quickly as possible is *paramount.* To that end the amount of tutorialization—or any other sort of obstacle between the player and the game itself—should be minimized or eliminated entirely.

Radically Simplified Design

A close relative to Pick Up And Play—mobile games are played on small touchscreens and as such, need to make the act of playing as simple as humanly possible. There is very little real estate, on both the screen and in the player’s mind, for overwhelming complexity.

Combat

An unusual—but not entirely unheard of—addition to the endless runner genre. In CYBERBIKER, the player will do more than ride—they’ll fight. But we need to keep the the rules above in mind when designing combat.

**GAME LOOP**

1. Game Start
2. Biker takes off
3. Player navigates the road, the obstacles, and the enemies
4. Player takes damage
5. Player dies
6. High score calculated
7. Restart? … Y/N

**WIN/LOSE CONDITIONS**

As anyone familiar with endless runners will understand, there is no real way to “win” a game like this. Put another way, the only way to win is to lose. That is to say, on a long enough timeline, CYBERBIKER players will *always*lose. It’s just a matter of how long they can survive. The longer a player can stay in the game, the higher their final score will be.

**CONTROL SCHEME**

As a mobile game, special consideration must be paid to the affordances/constraints of touchscreen devices. Furthermore, when designing gameplay mechanics, features, and systems, decisions must be filtered through these considerations. The more complex the gameplay or feature, the more potential there is for something to be lost in translation as the feature is abstracted into a touch-based interaction.

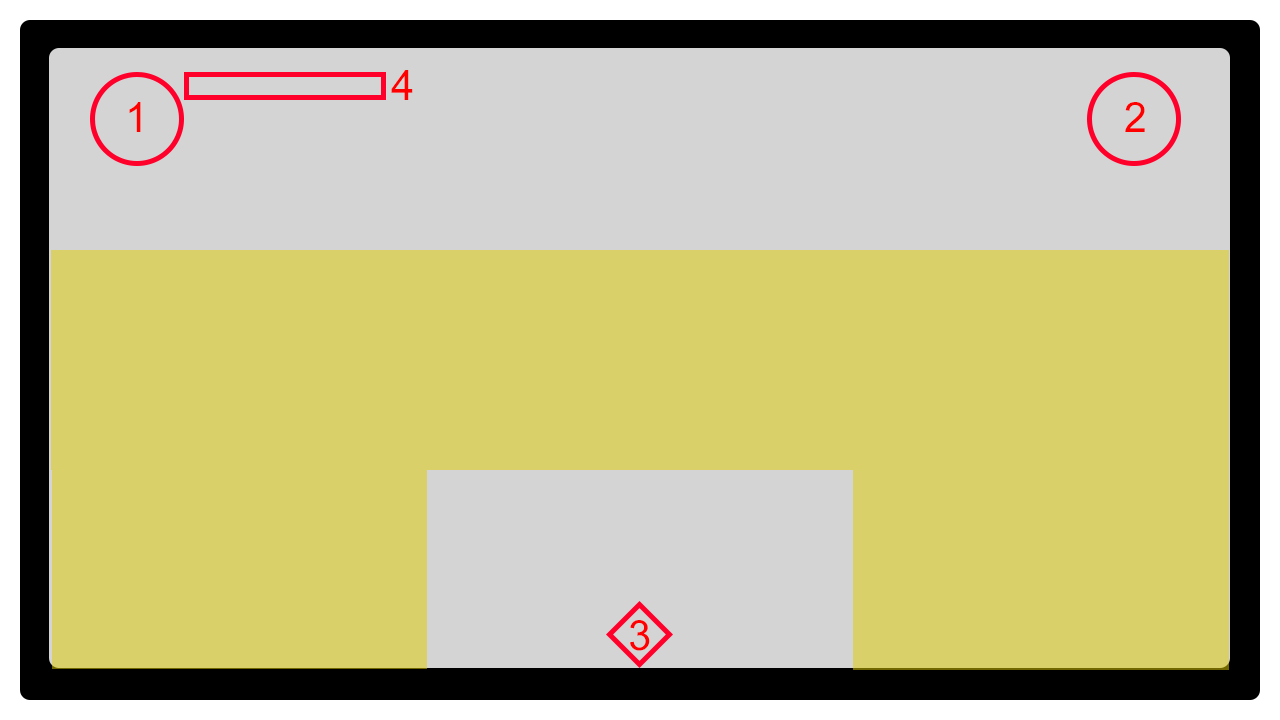
All the more reason to abide by our **Radical Simplicity** design pillar. This game is played by touching the screen in simple, gestural ways. The more the player has to learn in regards to these gestures, the more difficult (that is, “unfair”) the game will become. More on specific controls below.

**UI**

As with **Control Scheme,** special consideration must be paid toward screen real estate. The more items in the HUD (+ buttons, on-screen prompts) the more of a drain on the player’s attention bandwidth.

We can’t know how players will play the game (specifically, we don’t know if they are left handed or right handed or how they will arrange their fingers on the screen) so best effort must be made to make the experience as ambidextrous as possible.

We also need to free up as much of the screen as possible and reduce the total amount of buttons.

Below is a potential early stage UI layout.

**1 - Queued power-up. Tap to activate.**

**2 - Pause menu.**

**3 - Sword button.**

**4 - Meter (can be used for ammo, cooldown, etc.)**

**CORE MECHANICS & SYSTEMS**

Traversal

Forward movement is automatic. Moving side to side (and potentially up and down is up to the player). Side to side movement is enacted by dragging the biker to the right or left. Full range of motion from right to left is allowed.

Player Health

The player has a certain amount of HP. When they take damage, that HP goes down. When that HP goes down to 0, it’s Game Over for the player.

Shooting

By tapping on certain on-screen objects (enemies and perhaps specific obstacle types) the player can shoot their weapon in the direction of their tap. Certain constraints need to be placed on this mechanic (otherwise, the player could just snipe everything from a distance). An obvious constraint is ammo. Another might be the need to hold down a tap for longer to take aim or launch a powerful projectile. Another still might be an enemy type that is impervious to this kind of attack.

Sword Strike

In certain circumstances, the player can slash at their enemies. The player can also use their sword to deflect incoming projectiles. Constraints need to be applied to this form of attack as well, for the same reasons as above.

As for controls, the sword can be drawn by pressing an onscreen button. The player can then slash with it by tapping the appropriate object or enemy.

Power-Ups

Throughout the course of play, the player can pick up certain objects that grant them additional abilities or buffs. For example, the player may obtain a temporary invincibility power up or an upgrade to their bike or weapons.

**ENEMIES**

Before designing a whole stable of interesting and discrete enemies, we must first drill down into the *categories* the enemies will come to occupy. Once these categories are explicated, a single archetype for each category should be designed—in broad strokes—and then implemented in-game.

There are two good places to start vis a vis these categories: ranged and close-quarter.

Ranged

These enemies approach from in front of the player and hold their distance so that they can launch a ranged attack.

Close-quarter

These enemies can approach from in front of the player or from back-left or back-right (in relation to the player) in order to attack with a melee weapon or their vehicle itself.

**WORLD GENERATION**

The environment in CYBERBIKER is procedurally generated. This means that every run is different than the last.

**OBSTACLES, INTERACTABLES, & PICKUPS**

Combat is one half of the challenge coin in CYBERBIKER. The player must also contend with the act of traversal itself (such is the nature of most endless runners). Since forward movement in CYBERBIKER is automatic, side-to-side directional control and interaction with the game environment are the main concerns for the player.

As with enemies, creating a series of categories or buckets to use as guiding principles for design and implementation is key when it comes to this aspect of the game. To that end, the three categories to consider are:

Obstacles

Simply put, these are objects that must be avoided by the player. Coming into contact with them will damage the player, thereby reducing HP.

Interactables

These are objects that the player can interact with in certain ways, usually to their benefit. For example, a boost pad (the player drives over them to receive a speed increase that allows them to smash through objects or jump longer distances) or a ramp (the player drives over them to reach higher areas).

Pickups

More often than not, a power-up, but can also include things like ammo or additional health. These pickups can be obtained by running into them with the bike, but sometimes they need to be shot (if they are out of physical range).

**DIFFICULTY & PROGRESSION**

It is not enough that the player simply navigate obstacles and fight enemies. The level of challenge these actions present must increase with *time*. As the game progresses, the reaction time required by the player must narrow, new challenges must be introduced, and additional complexity must be introduced.

It stands to reason then that as challenge increases so too does the potential for rewards. This is to be reflected in the player’s score and the various rewards the player may encounter during their play through (currency, achievements, and so forth).

As what “shape” this challenge progression will take, that will be revealed through careful testing and balancing. But the variables required to increase that difficulty must be made available in-engine before that testing and balancing can begin. A good place to start is player speed (more on this **MVP**).

**SCORING**

Much like the shape/curve of challenge and progression, the “equation” used to determine a player’s score will be tested and iterated on over time.

To begin with, score must accrue at a base rate with time (meaning, if *all* the player did was travel in a straight line—if such a thing were possible—they would simply be rewarded for *existing).*

On top of this base layer, we will then need to add points for successfully navigating obstacles, defeating enemies, and pulling off difficult maneuvers or finding secrets. And on and on this list will go as new features are developed…

**MONETIZATION & IN-GAME SHOP**

The player will pick up in-game currency (IGC) during their run. They can also be awarded IGC after having completed a run. Finally, they can purchase ICG directly through the game’s cash shop.

As to what IGC will be spent on, there are of course many options, but as with many things we’ve gone over above, there are broad categories to consider.

Non-Competitive Benefits

These purchasable items do not affect the player’s performance. They will most likely be cosmetic in nature (new skins, for example).

Competitive Benefits

These purchasable items will give the player an edge in some way. For example, the player may purchase a resurrection in the shop that will activate upon death in-game, allowing them to continue their run.

**ACHIEVEMENTS**

With “simple” mobile games that have relatively short runtimes, it is important to encourage return visits to the game. Of course, high scores are one way to do this. Another is a robust achievement system.

Not only are achievements a “cheap” way to reward the player for engaging with the game in a deeper way (you’ve killed 1 enemy, now try 5), they also encourage the player to experiment or discover new parts of the game (use a ramp to reach the upper level of the game’s environment).

**MINIMUM VIABLE PRODUCT (MVP)**

The CYBERBIKER prototype is in good shape. In order to take the next step in development, the following steps need to be taken:

* Implement basic scoring system
* Implement health system (and print to screen)
* Implement basic UI
  + Score
  + Health
* Implement Game Over conditions
* Implement basic challenge/progression curve, ie. increasing speed/decreased reaction opportunity
* Implement basic combat system
* Implement simple ranged enemy
* Implement simple close-quarters enemy
* Implement one interactable
* Implement one pickup (a power-up)
* Basic world generation

**OPEN QUESTIONS**

* Tutorialization?
* Constraints on sword—time?
* Can the player bash into enemies and run them off the road *Burnout*-style?
* How will coins work? What are they used for?
* What triggers a change in biome with world gen?