Bike Messenger: Design Document

Luke Daschko & Morgan Lay

The Instructions

The game is played using the ‘A’ , ‘D’ and ‘J’ keys, which control left-right movement and jumping respectively. The player interacts with the game using their mouse and the keyboard. There are on-screen instructions which tell the player how to advance to the next screen. When the player gets to the level-select screen, they interact with the game by clicking the area they wish to venture to. Additionally, when players arrive at the upgrade screen they can interact with the upgrading via clicking the buttons beside the desired upgrade. Players can pause the game during gameplay by pressing the ‘P’ key.

The goal is to keep Dude going as long as possible until either he falls, or gets to the end of the level. Maneuver Dude side to side using the ‘A’ and ‘D’ keys dodging obstacle in his way; he isn’t slowing down for anything. If Dude cannot get around an object, his only chance is to jump using the ‘J’ key! Struggling to get through the game? Perhaps press the ‘P’ key to pause and take a breather. Get to the end to help Dude arrive on time!

The Strategies

Pre-Created Levels

The pre-created levels give the opportunity to player to choose one of two strategies to complete the level. The first strategy is a more safe, and less risky one. This involves following the path of least resistance in the level. In both levels there is route the players can take with minimal presence of any objects. This will ensure that the player reaches the end of the level safely with no hassle. However, due to Dudes larger distance from the obstacles, the players will gain a very low amount of Punk Points, disallowing the player to upgrade Dude’s abilities.

The second strategy is to follow the route of most resistance. This involves moving Dude to as many obstacles as close as possible without actually contacting them.This strategy is more risky and may lead to the player losing a life or losing the game. However, this strategy also richly rewards the player with Punk Points (as they would collect a high amount being close to the obstacles). The Punk Points can then be used to upgrade Dudes movement and jumping abilities, essentially making the levels easier to run through.

Infinite Level

In the random generated level, the basic strategies is to simply survive as long as possible. Unlike the pre-created levels, there is no benefit to moving closer to objects, and as no Punk Points will be rewarded for this action. A more complex strategy would be to aim to never be too far left, or to far right. Having Dude moving in these areas is risky as it can trap Dude. When Dude is at the far left of the screen he has no direction to go in but right, minimizing his variety in movement options.

The Idea

The main idea behind Bike Messenger came from our idea for a points system, later called Punk Points. The points are gathered based on whether the player is within a short distance to an object. The game inspires players to play riskier and rewards risky play with punk points, which can be used to upgrade the players movement and jumping. This interesting concept is what spawned Bike Messenger. We implemented this idea alongside an upgrading system to the skeleton of a top-down runner game, and added our own twist on it.

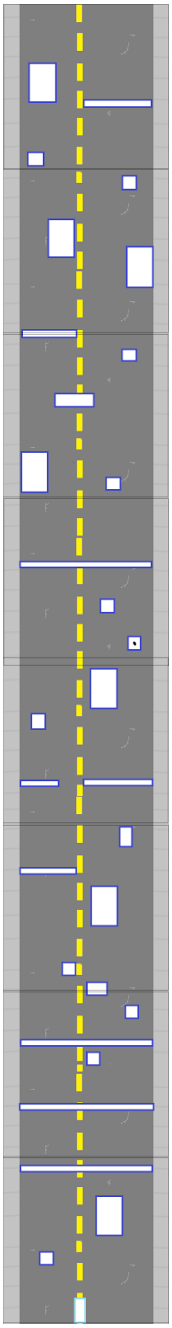
Ultimately when designing levels for Bike Messenger we felt that it wasn't enough to have set levels, so we had the idea of adding an endless level. To add a high level of variety to the game, the new game mode was added with new goals and new ways to enjoy the game. The infinite level introduced a familiar gameplay from more traditional top-down runner games: survive as long as possible. The infinite level gives players something new to do in the game. The way players approach each game-mode is familiar yet different, so it's an interesting feature for a game.

The Fun

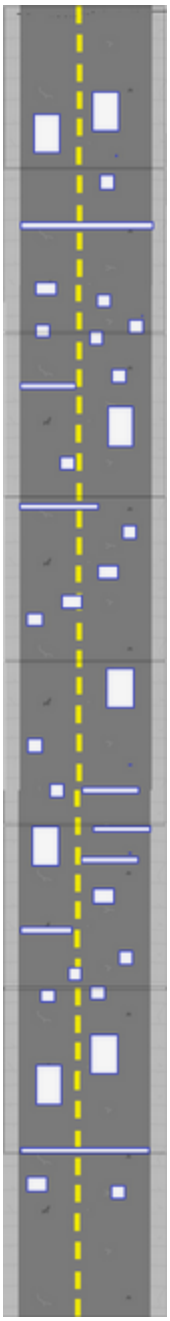
Our game was designed to have replay value and be exciting and different every replay. First let's focus in on the maps. For our endless level, the background does not change but it is relevant to the game and adds to the player being immersed in the character Dude. It scrolls by to simulate forward movement from the players perspective. The obstacles used change colours often to lessen the feeling of repetitiveness. The random nature of the spawning makes the experience different every replay because objects don't spawn in set positions, but rather of a random type and at a random position. As for the pre-set levels, we opted with the same background, same objects but placed them in a fair way so players of different skill levels could enjoy it, while offering challenging paths for experienced players.

The Level Designs

Level One



Level Two



The Game Logic

Game States

To switch game states in Bike Messenger we implemented a switch into the draw function that changes cases based on a game mode variable. In each of the cases is a different screen we wish to display when the game mode changed. For instance when on the title screen and the player releases any key, the game mode changes and the instruction screen is drawn. We worked the coding around this from the start because it seemed the most efficient way of progressing through levels and screens. There was a point in the development of the game where we had levels that went in sequential order, but we found it more exciting to switch levels and game modes via a level select screen.

Level Select Screen

The level select screen may appear straightforward and simple, but it ties together a lot of what makes Bike Messenger great. Think of this screen as the main hub for Bike Messenger, when you die or are done with an upgrade, you go back to this screen. Here you can select which level you would like to play, your current lives, and your best and last time on the infinite level. As for the logic when a players mouse coordinates are between the x and y coordinates of the top, bottom, left and right side of the buttons and mouse clicked is true, then the game mode changes to whatever the desired level is.

At the level select screen the lives system is shown on the bottom. Heart images, both empty and full, are loaded in setup and depending on a lives variable, displays the correct number of lives remaining for the player. The lives variable changes every time a collision is detected, and is visually updated at the level select screen. It displays images in set positions, and if lives does not equal either 3 or 2, shows the respective empty heart.

The best-time is calculated and recorded here on the level select screen. It works with a simple loop that checks whether the last time is greater than the best time, and if it is, then it replaces it. Because the number is in millis, calculations are done to display the appropriate time in terms of minutes and seconds. The number is taken, divided, rounded, multiplied and subtracted from the total to determine the next smaller increment of time.

Upgrading

The upgrade system works with global level variables that initially are set to 1. These variables are multipliers for the speed, and jump time. The global variables only change when the player presses the button on the upgrade screen and has enough Punk Points to cover the cost of the upgrade. When an upgrade is purchased, the level increases. Next time the player plays a

level, the level is multiplied by 10% of the base speed/jump and added to the speed/jump to increase the value. It works by using if statements to check whether the player can purchase it, and whether an upgrade has been done.

Pausing

Implementing a pause feature was easier said than done. It works by pausing the draw loop and resuming it when unpaused. Instead of switching game-modes, the pause feature is within the code of the levels. When a key is pressed, it checks to see if the key pressed was the 'p'. If the 'p' key was pressed, then it checks using if statements whether the player is playing or is at a menu. If the player is in game, it checks to see if the player is already paused. If the player is not paused, the noLoop() is called along with millis, text and filters. The text and filters execute one and show that the player is paused and how to unpause. Millis needs to be called because our velocity works based on it, so if noLoop is called and the time paused is not recorded, then the game will glitch and will have continued going when the player was paused. If the player presses the 'p' key again, then the game checks to see all the above if statements. The game doesn't continue to draw the filter and text, loop is called along with millis to check the time difference so that the game resumes where it left off.

Pre-Created Levels

The pre-created levels work in five subsections: object spawning, punk points, dude movement, background animation, and collision detection.

Object spawning works by holding all objects for the selected level above the display window, and adding velocity to a set at certain times. All obstacles are stored in an array for each level. each obstacle in the array is stored as a "obstacles" object. In the level() function, a for loop loops through every object and checks its phase attribute. Depending on what its phase is the function will add velocity to all objects in the current phase. The current phase is decided with the for loop by checking which objects are have the attribute flag equal to true. If any object with a flag = true, passes the y-value 250, the phase increases by one. Also to note is that car objects have different velocities than other objects, to give the illusions that all objects but cars are not moving and that they are passing you because you are moving, but cars are moving, and so their velocity is 330 rather than 300.

Punk Points work by checking if the dudes distance is close enough to receive Punk Points, and if he is, the player continuously earns the Punk Points until the player is no longer in rang to receive them. In the level() function, the for loop that loops through each object, there is an area of code that is dedicated to checking for Punk Points, which involves calling the function PunkPointsChecker(), and if and only if it returns true, the code continuously adds Punk Points. PunkPointsChecker() works by creating two float variables, dist_y, dist_x. These two

variables essentially calculate the distance from dude to each object in the y plane and x plane. Next, the function creates two more variables y_dist_for_points and x_dist_for_points. These uses a (complex) formula to calculate the minimum distance in order to receive points. The reason this method was used over the built in distance method Processing offers is due to the fact we wanted to manipulate how the distance is check. We wanted to check if the player was next to an obstacle not the distance from the objects center vector. Finally if the the two distance values are greater than the minimum distance values the function returns true.

Dude's movement is simply done by checking when the certain keys are pressed and simply velocity formulas to calculate the speed of the movement. The function keyPressed() and keyRelease() record when the keys A, D and J are pressed and change the values of pressedL (move left), pressedR (move right), and pressedJ (jump) to true or false accordingly. The controls() function does the rest of the work. When pressedL or pressdR equals true this function adds an image of the dude moving left or right and manipulates the the variable dude_x which holds the x value of the FBox which represents Dude. It multiples the set velocity times the jumpspeed (which is used to slow down movement when jumping) and by the movementLevel which makes the dude move faster based off upgrades. It is also multiplied by d_time, to ensure that frame rate issues won't damage the game severely. The jumping is a little more complex. The jumping works by setting a timer for the dude to stay up in the air for a certain amount of time then return to the ground. The illusion of Dude jumping is created by going through and a rray for progressively larger and larger images of dude, and attaching them in sequence to Dude before the timer, and stop at the highest point while the timer is running, then once the timer is over the images will regress in sequence.

Background animation works by constantly looping two images over and over in the background. Once the first image has hit the y-value of 814, the next image is moved above the display window and moves down until it hit the y-value and so on.

The collision detection is a method from the fisca's library and therefore most of the work is down without the programmer knowing. What we can explain is that when a collision happens between the Dude and an obstacle, the gameMode changes and a life is lost.

Infinite Level

The infinite level works by creating obstacles and having them display. Creating the object is broken up in four steps. Step one is to choose a size for the object, this is done by randomly producing an index value for an array that holds set of vectors. Each vector has a height value and a width value. These height and width combination in the array are pre-created from the list of 5 object sizes. The next step is to choose a position for the obstacle object. For the x-value this is done by randomly generating a number from 50 to 450 minus the width of the current object. This ensures the object will stay within the 400 pixel wide road. For the y-value this is done by randomly generating a number from a set of intervals starting at -620 to -650 and

increases by 100 each every time a new object is created. The final step is to simply create the instance of the object. Objects are made in sets of 8, and once 8 objects are made the y-value interval resets by to -620 and -650. The program is notified when to start the next cycle of object creation by checking when the last object passes the y- value of 500. A timer is also include that records how the the player has been playing the infinite level. With this information a timer display is created, notify the player of there time, and the program also uses the timer info to increase the velocity of the objects by 200 every 20 seconds (or 20000 milliseconds). The Dude movement is exactly the same as in the pre-created levels.

Balance

Upgrading:

Upgrading works by adding multipliers to movement speed and jumping duration. Due to the nature of how the upgrading works, a great deal of effort was placed into making the game enjoyable and balanced. The first thing we had to consider when implementing the upgrading system was how much much the upgrades would cost, how much they would affect the player, whether it transfers over to the infinite level and how additional upgrades stack with current upgrades.

When choosing an appropriate cost, we had to determine what amount of punk points was a reasonable amount to assume players could accumulate each level. After determining that 500 punk points was a reasonable cost, we played around with the multipliers and found that a roughly 10% increase of the base speed works best. We found that anything more quickly stacked up and became a disadvantage for the player. After playing around with both of those, we determined that half of each upgrade would go towards moving while in the air to balance jumping. As for transferring over upgrades to the infinite level, we found that we wanted to keep base levels for it because it would undermine the highscores if players could grind the Punk Points to do well in the infinite.

Level Design:

Infinite Level

When it came to the infinite level, we found a lot of issues with balance because of the random nature of the spawning. Infrequently objects would spawn in ways in which players would have a very tough time avoiding. After discussing it and playing the game a lot, we found that it wasn't a deterrent to the game. Most games have obstacles or scenarios in which players cannot escape, and sometimes the same is true in our game. Other issues with balance include upgrade system transferring over, but that was discussed at length in the above paragraphs.

Finite Levels

In the finite levels we had to design the maps in a way that was both fair and challenging for players of different skill levels. The map designed with the idea in mind that there's two ways to complete each level: by follow a path with little to know object presence and a path with extreme object presence. This was done by ensuring that through most of the level there was never a moment where players were faced with nothing but objects everywhere, and were never faced with a lack of objects, but a geographical divide of one side having many objects, another side have less objects. This ensures balance in each level as it allows a multitude of strategy. Maps feature a low-risk, low-reward path and a high-risk, high-reward path.

The Kleenex Test

Our Kleenex test showed our expected results. After allowing the tester to play all levels and have multiple tries at the infinite level, they proceeded to say that our game, although was fun to play was not perfect. The main concern from the tester is a lack of content. The tester explained that they would have liked to see more created levels, and more variety in of obstacle images. We both acknowledged these concerns, and if we had more time would have have certainly implemented more levels, but these took a very long time to design, then map out, the code. The concern of a lack of visual variety was also an expected complaint. During the creation of our game the aesthetics were never a focus for us. We did however focus on visuals but almost entirely through the lens of clarity. Therefore adding more variety in visual design of the obstacle was never something we aimed to score high in. Other than this the tester explained how the controls were fluid and the bug with the jumping was bothersome, but manageable. The bug with jumping happens when the player jumps over and object and although Dude appears to be jumping (through the illusion of increase in size and decrease in size), Dudes image fails to overlap the obstacle. This is a bug we had from the start and never seemed to be able to fix. The tester said their favourite feature was the upgrade screen. The tester explained “I like that I get to upgrade my dude after the work I do in each level, that’s cool”.

What game design elements are involved?

Clarity

Bike Messenger produces clarity in many aspects. In term of controls, Bike Messenger successfully communicates to the player what their actions do to Dude. When the player hits the A button the dude object attached an image of Dude moving left, making it clear that the event of

moving left is successful. The same applies to when the user presses the D key, but in terms of moving right. When the player hits the J key Dude's image size gradually increases, stays at the maximum size for a short while, then gradually decreases size to the original size. The gives the illusion that Dude is leaving the ground then returning back to it. This provides a good amount of clarity to the player that when J is pressed the jumping event is in fact successfully being executed. The event of receiving Punk Points is also well recognizable as the number of Punk Points increases when the player is in range of receiving Punk Points. This is a good balance of clarity as any more visual cues would risk becoming a distraction to the player. In terms of players knowing their goals, this is ensured by the pre-level functions. What these functions do is display a screen of text the notifies the player what the object of the stage is. For both pre-Created levels, the this text tell players to reach the end without dying and notifies them of the benefits of Punk Points as well as how to obtain them. For the Random levels, the game displays a brief text notify player to lay as long as they can and to try to beat the highscore. This clearly and cleanly notifies players of their current goals.

Variety

In terms of variety, Bike Messenger gives a fair amount. The main area where variety is shown is in variety of game types. Bike Messenger includes to different styles and types of games in it. One is a level based upgrading runner, while the other is a high score based infinite runner. These two clash in game types and give Bike Messenger a rich sense of variety. Rather than placing focus of variety within enemies or objects, we decided to place a powerful amount of variety with in style of gameplay and goals. We wanted provide the effect of variety, freshness and the feeling of something new, to the player through these lenses. This is why we created an infinite level, and then pre created levels. In the infinite levels player are exposed to gameplay were pure survival and stamina is required. In this mode players also stride to beat a high score, which encourages a different kind of gameplay and gives a new kind of fun to the game. The created levels offer a different experience altogether. Rather than focusing on pure survival or beating a highscore, in this mode player focus on completion of levels, and improving Dude by collecting points. This mode brings new experiences to the player, and changes gears on goals. We felt that by injecting variety in the game modes rather than in character design or object interactions, we could provide variety in different way, but still equally effective.

Innovation

Bike Messenger shows off its innovation with the Punk Points system. The Punk Points system was the root for how we created Bike Messenger. When we were thinking of ideas we came up with the innovative idea to create an arcade style runner game, but giving it a twist. Eventually the idea was set to create a runner game where the player is rewarded with points

when traveling next to obstacles, which he can then use to buy upgrades, and make the game easier. In essence the idea was to game the game easier, by playing a more risky gameplay style. This idea seemed really interesting and exciting to us, and from what we understand few games mimic what we have implemented. Another area of innovation is the theme of bike messengering. The only other well known bike messenger game is Paperboy for the NES which is different as it more involved delivering newspapers to houses right next to each other, and a bike messenger is not a paper delivery boy, but similar. Bike Messenger culture has been taped in film with movies like Quicksilver (starring Kevin Bacon) in the 80s and more recently with Premium Rush (starring Joseph Gordon Levitt), so we thought it was time for someone to try using this theme in video games.

Fiction Summary

You play as Dude, an eleven and a half-year old bike messenger. He is an experienced biker and can move effectively and jump for extended periods of time. Dude has an exciting life outside of being a messenger. For fun he can be found biking, mountain biking, race biking, stunt biking, marathon biking, and satisfying his insatiable hunger for anchovy pizza! In the story Dude is caught up in detention and rushes out biking so that he doesn't miss his date. Whether he gets there or doesn't, he falls short of getting the girl. He finds himself biking home frantically another day to get a slice of anchovy pizza. To his delight it was worth the trip. Above all else, Dude wants to appear like a badass so he prefers to stay close to obstacles because he's a punk. Sometimes Dude can be caught training for his big race the Bike Messenger Tour de France.

Extra:

Dialogue:

Level One *Intro*

After finishing a long half-hour work day,
your mom is calling you home for supper.
Normally you wouldn't care
because you're you're a punk, dude.
Tonight is different.

Tonight, mom ordered pizza, which means there's a hot 12'inch double anchovy pizza waiting for you at home!
Nothing will stand in your way.
Don't get too reckless Dude, you don't want anything cramping your cool.

Defeat

DUDE, that's a bummer. Now your pizza is going to be cold AND you've lost street cred.

Victory

Pizza err…. I mean victory has never tasted any better! Wicked moves Dude!

Level Two *Intro*

Why does the teacher always give me detention? I'm allowed to draw on the walls at home. Oh SH*T! I'm going to miss my date with Dudette! Punks don't go to detention anyways. I better hurry and roll up in style!

Defeat

Dude… I can't go now. She'll notice my scraped knee, that's not cool!

Victory

"Dudette, you're so pretty… uhh… I gotta go. I have to water my sunflowers." Phew! That was a close call, I almost caught cooties!

Infinite Level *Intro*

Time to begin my training for *Bike Messenger Tour de France*!

Defeat

Well that sure ended quickly, maybe I'll last longer next time!