

Game Design

Cometes Interface Design

Individual report

Written by Jonas Wæver

Teachers: Miguel Angel Sicart Vila and Douglas Edward Wilson

Supervisor: Christos Iosifidis

2680 words

Contents

Introduction.....	1
Cometes.....	1
Motivation.....	1
Concept development.....	1
The discovery problem.....	2
Design objectives.....	3
Art direction.....	3
Animation.....	3
Audio.....	4
Art iteration.....	4
Level design.....	5
In summary.....	6

Introduction

Cometes

Cometes is a physics puzzle game for the iPhone and iPod Touch platforms. It is based on three distinct core game mechanics utilising the device's touch screen and accelerometer input methods. It uses a linear progression structure where each level is one screen. The win condition in every level is to move your avatar, a small blue comet, to the exit, represented by an opening in the side of the level marked by a red fire animation. The obstacles are sine waves which destroy your avatar, sending it back to the level's entrance, teleporters which move your avatar somewhere else in the level, suns with gravity fields which attract your avatar, and humble walls that block your progress.

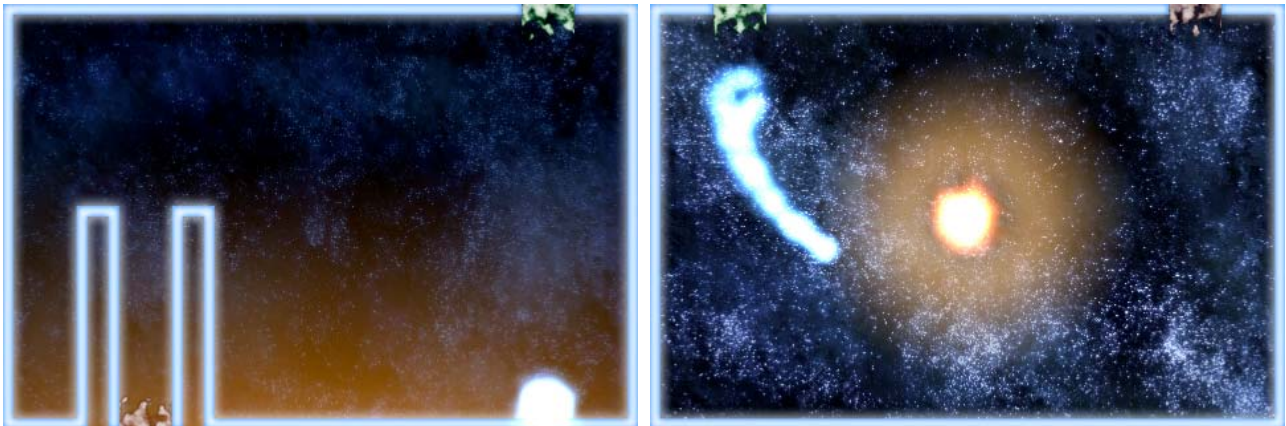


Fig. 1 and Fig. 2: Example levels from Cometes.

Motivation

I've chosen to focus on the interface design for our game, Cometes, both because it's the aspect I've been most involved in, and because it has posed (and still poses) some interesting challenges and problems. I consider interface design to be a very important aspect of game development in general, but especially in small casual games where there is little to no narrative with which to reward the player and maintain his or her attention. I have held the opinion throughout the project that the interface design for Cometes would be of critical importance for further reasons than these, and in this report I hope to explain why and to describe what problems we identified, and what we did to solve them.

Concept development

Our original design objective with Cometes was to foster a sense of discovery in the player. We wanted our game to call attention to the affordances of the iPod Touch and to encourage players to experiment with their interactions with their device and the game. Our idea was to create a game where each level would be a unique puzzle that framed the inputs of the iPhone in a new way, and where the challenge would always be in learning how to interact with the level rather than executing a skilful manoeuvre or overcoming complicated obstacles.

In our attempts to realise this idea, we drew inspiration from the constantly changing interactions of WarioWare, the exotically mysterious experimentation of Windosill, and the mechanics of a range of different iPhone games such as Spider or physics puzzle games like World of Goo. The design we settled on

called for a murky ocean of colourful bubbles that would need to be manipulated in completely different and surprising ways in order to unlock them so they could be strung together into a linear set of physics puzzle levels to be zoomed into and played through.

Fortunately we quickly realised that we lacked the time and resources to implement this concept, and we decided to focus on the physics puzzle layer. After a few weeks of work on the prototype, we had to admit that we would not have the time to create many completely unique levels, and that it would be wiser to select a few mechanics and use them for several levels each. After careful consideration, we chose a Spider-like flick mechanic, a slingshot mechanic reminiscent of World of Goo, and a tilt mechanic using the accelerometer.

The discovery problem

The major design challenge in terms of our interface was making sure the players would understand what was going on when we kept switching around the core mechanics from level to level. One of our earliest ideas was to change the look of the walls depending on what sort of obstacle needed to be overcome (for example, metal walls for the sine waves or blue neon walls for teleporters) and using the backgrounds to show what sort of mechanic was at play and which direction the gravity was pulling, if applicable (eg. star fields for tilting or a gradient for flicking). We decided against this solution because it would create a very chaotic and muddled visual expression.

We settled on not indicating which mechanic was used in a given level, hoping to maintain an element of the original aesthetic of discovery. Our hope was that players would quickly figure out how they were meant to manipulate their avatar by simply playing around with it until something worked. In order to introduce each mechanic properly to begin with, however, we added three very simply “tutorial” levels, where players were free to mess around and experiment without any obstacles or threats to worry about. Based on early user testing, we also added an arrow pointing to the exit in each of these intro levels, to make sure the player understood the goal of the game early on.

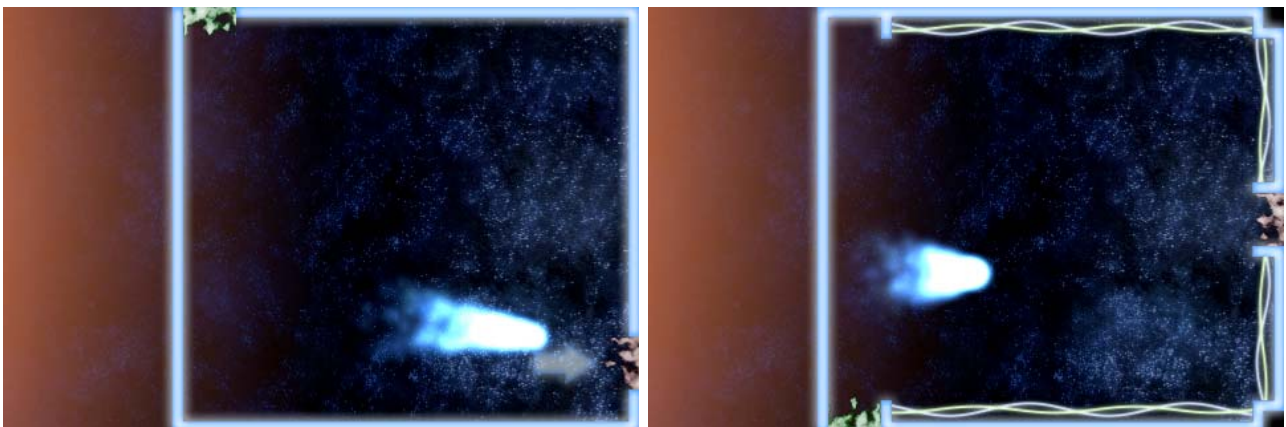


Fig. 3 and Fig. 4: Tutorial level and advanced level using the slingshot mechanic.

Design objectives

Inspired by casual games in general and PopCap games in particular, one of the major goals of our interface design was to make the game enjoyable to interact with on a basic level. Our stated intension was to create a “tactile” or “juicy” interface, which we hoped would both help the players to understand the consequences of their input and make the game more superficially rewarding to play. A tactile interface uses sound and animation to create the impression that the player is manipulating physical objects by interacting with the game. This makes the fundamental game experience more pleasurable, and if used correctly, it can also help to reduce the time it takes for a player to understand how his or her input is interpreted by the game. We immediately established two design goals:

- The players should always know that their input has been properly registered. If they touch their avatar, it should react to that touch. If they touch an object that does not react to touch, a sound might play to tell the player that the object is static.
- The players should be able to tell how objects interact with each other and with the levels. If the avatar bounces off a wall, it should be obvious why this happens and what the results are.

Art direction

We experimented with several visual themes early on, including translucent dark walls over an iconic gradient “sunset” background, and viscous bubbles in murky water. Ultimately we decided to go with neon blue walls on a star field, primarily because it was the option that looked the least cobbled together. This also had the effect of making our avatar (made of particle effects and originally conceived as “a spark”) look like a Comet, which gave us a name for our game. The theme we chose also fed back into the game design when it inspired us to add gravitational attractors in the form of suns.

Once we’d settled on the space theme, most of the graphics were straight-forward. We chose to use the sun particle effect instead of the black hole graphic for the gravity attractor in order to match the orange gravity gradient on some of our levels. Our sine waves were originally a more standard red laser beam, but this seemed too obviously man-made, and we replaced it with the more organic-looking wave graphic.

Animation

One of the first things we did to realise our design goals was to identify which elements we wanted to animate. We decided to use animation to call attention to the most important objects in each level, which meant animating anything the player could interact with or needed to avoid. With this in mind, we started by animating the level entrance and exit graphics, creating two fire animations with different colours and opposite animation sequences. As early as the prototype, we have been using the same particle effect for the player’s avatar, and we were quite happy with it, but it needed proper animations for entering or leaving the level, as well as for dying or entering teleporters. This was done in code. Another important object to call to the player’s attention was the deadly sine wave, which we chose to animate frame by frame in Photoshop like the entrance and exit. Finally, the teleporters needed an animation to make them stand out from the background, and we decided to simply overlay the graphic with a translucent copy of itself and program the overlay to rotate. We also animated several other objects for levels which were omitted from the game due to time constraints.

Audio

Another major goal was to make the audio and the visuals of the game support each other to form a coherent aesthetic. We called in some favours with a sound designer outside of the university, and he supplied us with the 27 sound effects that we requested. Our central goal with the audio design for *Cometes* was to make sure it was as pleasant and unobtrusive as possible, since many of the sounds would be played over and over and over throughout the game's many levels. Unfortunately, we had very little time to iterate on the audio, and due to our priorities, the sound effects didn't make it into the version of the game we've handed in.

Art iteration

The first problem early play testing revealed was that players really had trouble grokking the slingshot mechanic. The mechanic was originally inspired by certain levels in *World of Goo*, and the particle effect of the avatar was meant to stretch as the player pulled it back. Unfortunately the framework we've used didn't easily support such detailed behaviour in its particle system, and while we were working out a solution, we had simply omitted any kind of visual effect. Unsurprisingly, having to stretch your avatar like a rubber band and aim it without any visual feedback was unintuitive.

The first solution we tried was to attach a second particle emitter to the player's touch, but without any indication of the connection between the two points, it was just as confusing as before. Not long before our formal usability testing session, we rendered a simple line between the two emitters, which helped some players while others remained confused. Some of our play testers indicated that the problem was the identical appearance of the avatar and the touch emitter. Our final solution was simply to remove the second emitter but keep the line. Sadly, we haven't had time to test this configuration.

Our second problem was distinguishing between the player's avatar and the gravitationally attractive suns. Since we were using the same effect with different colours, some of our early testers expected that there would be some functional similarity between them as well. Further, a tester requested that we display the gravity field of the suns, showing that their influence extends beyond their centre point. Our solution for this was simple: we added large radial gradients to the background image of the relevant levels, aligned with the position of each sun. Judging from our play testing, this was a good solution, as none of our testers have had trouble identifying the purpose of the suns.

Our final change was to alter the colour of the avatar in each level depending on the core mechanic used. This was a solution we'd discussed from the beginning of the production, but there was some disagreement about the accessibility concerns it would introduce – colour blind players would probably be unable to distinguish between some of the colours, which was a special concern for us considering that Miguel is such a player. In the end, however, when play testing indicated that identifying the interaction method in each level was more of an annoyance than an engaging challenge, we had no time left to implement a more accessible solution, such as tweaking the behaviour of the particles or adding a graphical icon to the avatar. If we decide to keep the colour indication in future versions of the game, we plan to add a settings menu to the game where colour blind players can substitute our colour scheme for one that they can differentiate between.

Level design

Level design is of great importance to any puzzle game, and in Cometes the level design tasks were closely related to the interface design. Not least because our levels were designed and built in Photoshop CS4 and then implemented in the game code with the help of Photoshop's grid. More fundamentally though, our game concept called for each level to feature unique interactions, which encouraged us to consider our levels as interface problems as well as puzzles or challenges of skill.

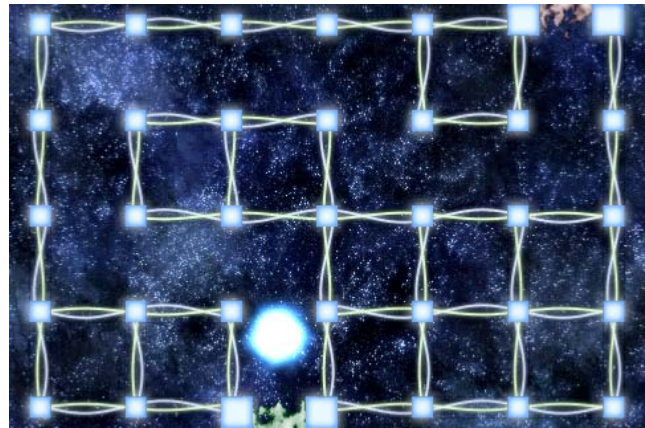


Fig. 5: *Easily* the most difficult level in Cometes.

Our level design process started with several days of brain storming. We would first have a meeting and talk about the sort of challenges and obstacles we might use, then part ways to create a set of level ideas each, meeting up again later to compare notes. When we had a sufficiently outrageous number of ideas and drawings between us, we compared the mechanics and objects needed for each of them and grouped them according to required resources. Then we picked our favourite level types to work on first. The features we chose to prioritise were the flick, slingshot, and tilt mechanics as well as the laser and teleporter objects. We soon added the gravity objects and a few associated levels to the list.

At this point, we still had several additional levels on the to-do list that required unique mechanics, as we were still working with the idea that each level should introduce at least one completely new feature. After a few weeks of prototyping put these ambitions into perspective, we decided to focus on variations on a smaller set of features, more like what one might expect from a traditional puzzle game. We tasked ourselves with designing 5 levels each, all using the same core mechanic and spanning a difficulty curve from the level that would introduce the mechanic to the level that would really test the player's mastery of it. Our design philosophy was to first introduce the feature in a completely no-pressure environment with no dangers and no obstacles, then step it up by introducing obstacles to be circumvented, and finally adding dangers such as sine waves that would destroy the avatar and respawn it back at the beginning.

Armed with these levels schematics, we created the necessary backgrounds and objects in Photoshop, and then we began to implement them in the code while the initial prototype of the flick mechanic was being further developed with the other mechanics and objects. We split the level construction between two of us, one finding the coordinates in Photoshop and the other typing them into the code. Thanks to the way we'd set up the level art in Photoshop, we were able to use production quality art work at this early stage without slowing down our iteration to any noticeable extent. This also helped us to more accurately play test the game much earlier than we would have been able to do with prototype graphics, which I think has been a significant boon considering how important our game's interface (and graphics) are to the user experience.

In summary

Inevitably, the game we're handing in is quite different from the game we set out to create. As our design documents will show, our original plans were much more experimental and unique, but also very poorly defined. As we struggled to distil our ideas and ambitions into concrete designs, the game predictably came a lot closer to something that I feel has been seen before. The game evolved from an unstructured play thing to a level based physics puzzle. I do feel, however, that the game we're handing in has retained a hint of that anarchic sense of discovery and surprise that we originally wished to create, and I believe that maintaining and strengthening this aesthetic while ironing out some of the ambiguity and confusion and other rough edges of the game will be our primary challenge as we continue to work on the game in preparation for releasing it on the App Store.