Updating Snake '97

About the challenges of developing a wildly popular game

Willem L. Middelkoop

Feb. 21, 2018

Few years ago my brother threw a beer on my iPhone in an Amsterdam bar. The poor thing didn't like the Dutch brew as much as I do: it died. While waiting for a new phone to arrive, I used an old one that couldn't do anything but texting, calling and... Snake! The idea for Snake '97 was born and this month it was time to update the wildly popular game.

Café 't Pakhuis in Amsterdam - where Snake '97 was "invented"

Snake '97

Originally I programmed the Snake game in 2011 for iOS. I took me about two weeks to create a working prototype that looked and felt like the true classic from the 90s. My other brother helped me with graphics by taking some sharp looking macros from my old phone (it's still a weird idea that so many of you have a copy of my phone in your pocket).

Snake '97 running on iPhone X - looking and feeling like the retro mobile phone game from the 90s

It got picked up by some big blogs that made Snake '97 rock the top rankings all over the world. Today it's being played all over the world, translated into 53 languages and available for iOS, Android, Windows, ChromeOS and MacOS.

Snake '97 rocking the carts with Angry Birds and WhatsApp

Need for update

In 2011 there was just one iPhone, the iPhone 4. As developer you could safely ignore Android and Windows Phone wasn't even invented yet. Life was as beautiful as a (lazy) developer could wish: just one screen size.

iPhone 4 - ahhh sweet developer memories when one (small) size did fit all...

Fast forward to today, Apple currently sells 8 iPhones and there are millions and millions of Android devices out there, all with different screen sizes, processors, memory and software versions. It's an understatement to call this a challenge...

To maintain the fun in Snake '97 I had to update the game engine at its very core. Somehow I had to make the graphics fit every screen in the world and accommodate for the differences in processing power and memory....

Prototype of Snake '97 running on various devices

One game engine to rule them all

The original versions of the game all had their own engine. One for iOS, one for Android, one for Windows etc. When I launched Snake 2k (the Snake II game variant), I simply added another set of game engines. In total I programmed 6 different Snake game engines in Objective-C (iOS), Java (Android) and C# (Windows). Keeping these engines running on new devices with different screen sizes, processors and memory configurations became increasingly difficult.

But this month I bit the bullet, big time. I redesigned and build another Snake game engine. But this one was meant to replace all the previous ones. It must run on iOS, Android and Windows. I wanted an engine that was ready for the future, for any screen and any kind of input (keyboard, touch, pencil, gamepad, joystick, you name it).

Programming the new game engine and testing it on an HMD Nokia 2 running Android

HTML5 Rocks

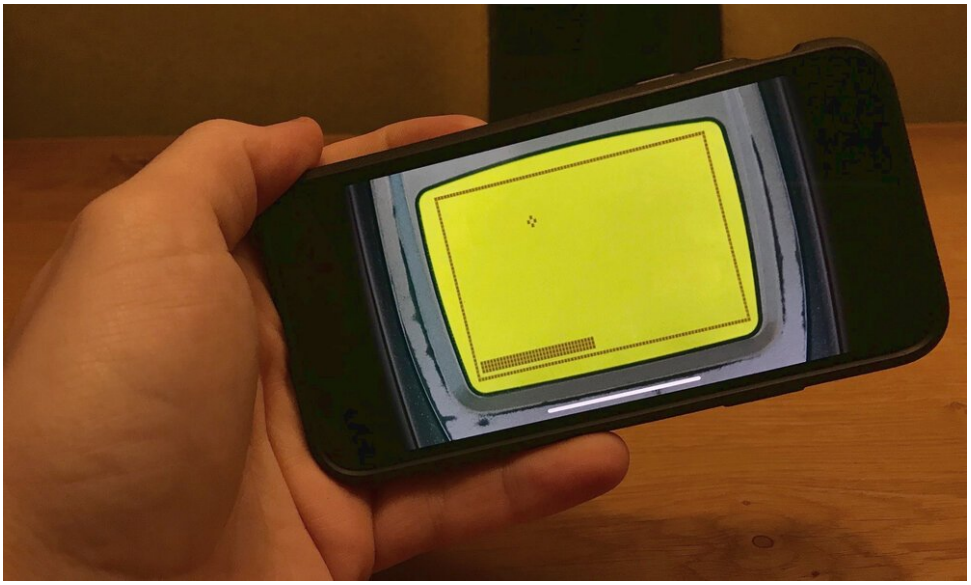
The web has caught up with native app technology. Many new API's have been introduced that enable rich interactive experiences on many different devices. I wondered how difficult it would be to make Snake '97 work on web technology. Only one way to find out, so I started.

Responsive graphics

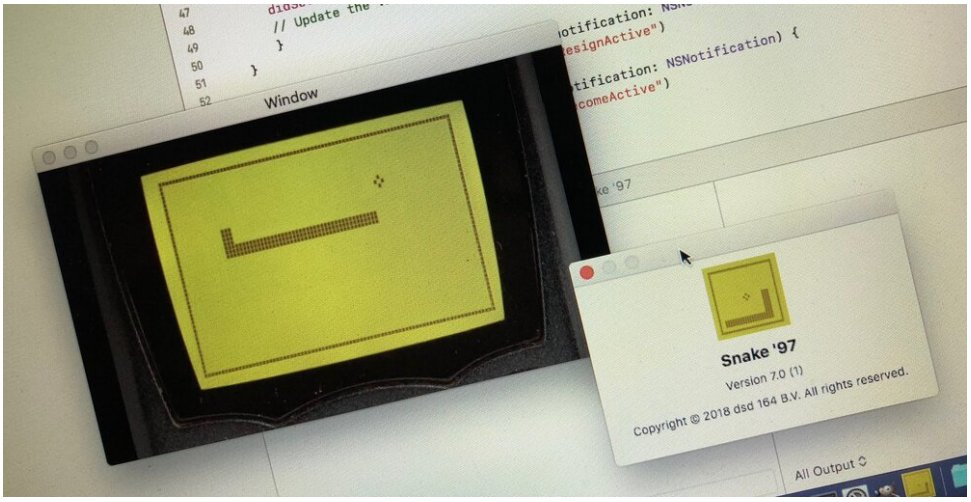
First I started building the graphics engine based on HTML5 canvas and a responsive scaling system that I designed for the Android version of Snake '97. In essence the engine measures available screen size, orientation and pixel density. It automatically translates high resolution photo material to match your specific screen.

Same 3310? Look closely and spot the differences between iPhone SE and iPhone X

The new graphics engine even accommodates for weird things like the iPhone X notch. This all happens in a split second. I designed the new graphics engine to respond to changes in screen size, too. Now it's possible to play the game in portrait or landscape, in a Window or full screen (or split screen if you're on iPad).



Wide screen gameplay is now possible thanks to the new responsive graphics

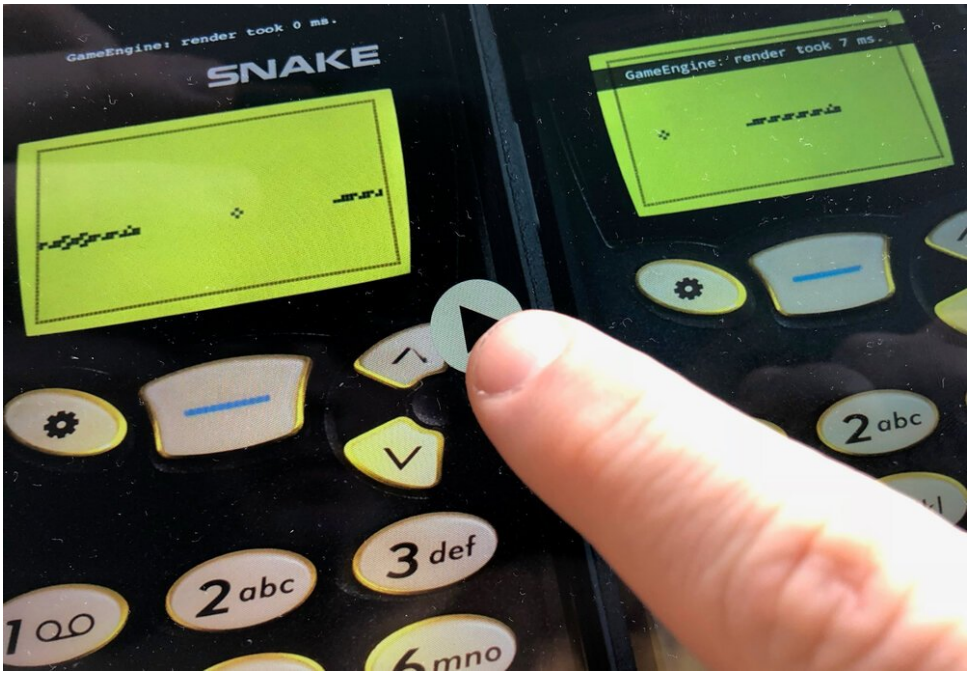


Snake '97 running as macOS desktop app made possible by the new scalable graphics

Differences in processing power

You think a simple game doesn't have to worry about computing power? Wrong! In a game like Snake timing is essential. People play the game with ”muscle memory”, repeating their yap-tapping on the touch screen without thinking about individual moves.

The game engine must be predictable, it must run constant. But moreover, the experience must be the same wether you're on an old Android device or a 6-core power house like the iPhone X.



Analysing gameplay speed differences using slow motion video

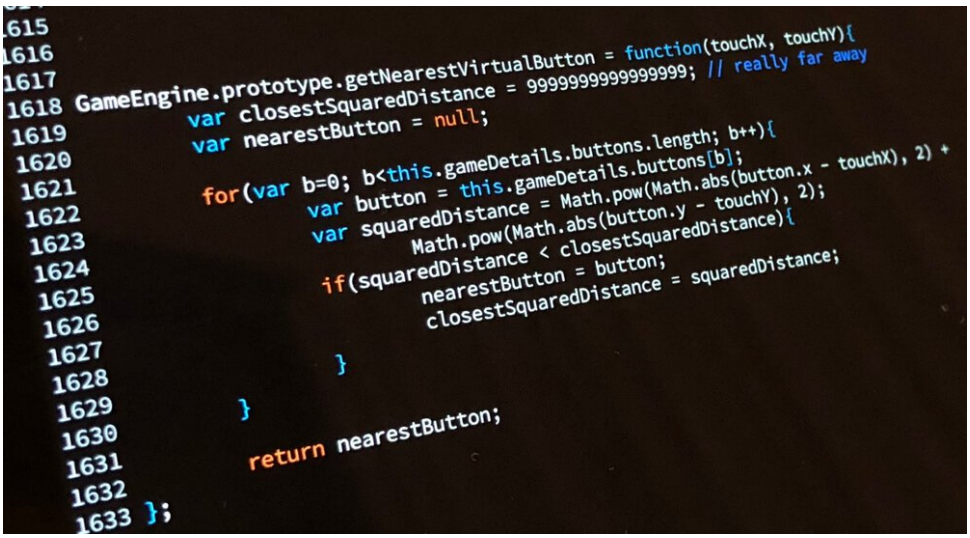
I used the iPad Pro's high speed video camera to capture Snake gameplay in slow motion. Using the video I could spot differences in speed, frame by fame. This led to an internal benchmarking system (using timers, counting milliseconds) that automatically adjust the speed of the game precisely tuned to your specific device.

Controls: Touch, Keyboard, Pencil, Gamepad

Players control the Snake using the retro controls (the old phone keypad). The original iOS version of the game used fixed transparent UIButton's that I placed over the photos of the old phone. A simple and effective start.

Unfortunately the buttons provided by iOS had some lag (causing unpredictable delays in input) and obviously didn't scale to different screen sizes. I literally hard coded new coordinates for every new iPhone Apple delivered over the past few years.... not a fun job.

The new game engine uses a trick that I developed for the Android version: dynamically positioned virtual buttons. Using the scaling information that the graphics engine calculates, I dynamically maintain a matrix of X and Y coordinates matching the virtual buttons. When a player touches the screen, the engine calculates the nearest button using mathematics, the Pythagoras theorem.



```
1615
1616
1617
1618 GameEngine.prototype.getNearestVirtualButton = function(touchX, touchY){
1619         var closestSquaredDistance = 9999999999999999; // really far away
1620         var nearestButton = null;
1621
1622         for(var b=0; b<this.gameDetails.buttons.length; b++){
1623             var button = this.gameDetails.buttons[b];
1624             var squaredDistance = Math.pow(Math.abs(button.x - touchX), 2) +
1625                 Math.pow(Math.abs(button.y - touchY), 2);
1626             if(squaredDistance < closestSquaredDistance){
1627                 nearestButton = button;
1628                 closestSquaredDistance = squaredDistance;
1629
1630             }
1631         }
1632         return nearestButton;
1633 };
```

The actual code from Snake '97 handling distance calculation using the Pythagoras Theorem

This works like charm and effectively makes the entire screen a button. The game is much more fun to play this way. In addition to touch, I added support for the keyboard so that people without a touch screen could play Snake, too.

While I was working on the controls I discovered some more HTML5 goodness, the GamePad API. It enables programmers like me to support gamepads, joysticks and other input devices connected through USB or Bluetooth.

Controlling Snake using a Super Nintendo gamepad is awesome (using a SuperSmart-joy USB connector)

Supporting different gamepads is challenging, but fun nonetheless. I for sure lost an entire day while ”testing” ... perfecting my skills using the formidable SNES gamepad and lovely analog stick from the DualShock controller. (”Sorry darlin, I'm working here...”)

A trick the original Snake couldn't: controlling the game using a Bluetooth DualShock PlayStation controller with an analog thumbstick

Internationalisation

If you read this, you probably understand English. But for a major part of the Snake '97 audience is non-English. In fact, there are players with more than 30 different languages. From Spanish to Russian, from Arabic to Hebrew, from German to Japanese. You name it, and Snake has it.

Designing an interface that relies on text is challenging if you aim to serve the world. Icons aren't flawless either (different meaning to an image). As UX-designer you have to think of concepts that are broadly used world wide. For Snake, I used an icons of a clock, speedometer, speaker and a trophy. These icons are common all over the world and where chosen precisily for this reason.

UX-design for an international audience. Designing interfaces without words is difficult!

The app description consists of text that I had translated by a professional translation service. Some other publishers rely on Google Translate - but I know that leads to dismal results.

World wide ranking

Back in the 90s you got yourself some good street credit if you've achieved a nice score in Snake. In the original versions of Snake '97 you only had your own highscore to beat and on iOS you could upload your score to GameCenter.

For the new version I wanted a more global reach, introducing ”world wide ranking”. It's the new integrated leaderboard that instantly ranks your game, comparing scores from people all over the world. There are three rankings: all time, 24 hours and last hour.

World champion of Snake '97 - but not for long...

The high scores are processed by an incredible 48-core Intel XEON server with a boatload of memory. It runs Debian GNU/Linux, MariaDB, NGINX and NodeJS. During peak hours it handles more than 10 game scores a second. The server setup is worthy of a dedicated blog post that I will write later - so keep an eye on this blog if big data is your thing.

This is what the cloud looks like, dear boys and girls.

Download now

There is only one way to find out if my work was any good: try it yourself.

- Android
- ChromeOS
- iOS
- Mac
- Windows
- Web

Please do let me know if you come across any bugs, you can find my contact info here.

Snake '97 available as free download for iOS, Android, macOS, Windows and ChromeOS