Subject: How to FAIL to make a Game on the Apple IIgs
Posted by Oz on Fri, 03 Oct 2014 17:02:41 GMT
View Forum Message <> Reply to Message

There are probably a lot of good advice about how to create a game on the Apple IIgs but I thought it would be also interesting to see what kind of errors people usually do when they want to create a video game and how they could be avoided.

Unlike the Apple IIe, the Apple IIgs is a complex computer and create a game on such machine is far complex than doing it on the IIe.

The first think to consider when people speak about creating a game is to understand what this means. To build a game, we need :

- A Game Designer. This a a Creative job. => He has the game vision. He creates level design (rules) and make sure the game is enjoyable.
- A Programmer. This is a Technical job. => He masters Assembly language programming and knows very well the low level of the machine (graphic, music, sound, OS, disk access) .
- A Graphist. This is as Artistic job. => He draws backgrounds, create Sprites and check animation steps to have a smooth animation.
- A Musician. this is an Artistic job. => He does the music theme and the sound part

On the Apple IIe, you could find one guy doing everything. Mostly because the Graphic and the Music parts were pretty basic (Lode Runner, Choplifter) . On more complex games, like Karateka or Prince of Persia, the guy (Jordan here) had to be good in every domains, and this is rare.

On the 16 bit period, very few people were capable to address the 4 jobs. Eric Chahi (Out of this world / Another World) was one of them. Most of the time, game were created by small team (1-4 people).

If you want to create a game for the Apple IIgs, you are probably already a programmer. If you hope to create something completely new, you have to deal with game design (what the game is about), the graphics, the music (including sounds) and finally the programming part. This is HUGE for one single person. Of course, the fun part of the job, for a programmer, is to program the game engine. Everything else is seen a time wasting.

The game design is of course the most important thing to end up with a good game. If you have everything else, you end up with something like Bouncin' Ferno which is very nice but boring to play. If you don't have a graphist, you end up with Star Wizard, a nice 'technical game' but the kind of game you will never try 2 times.

As a programmer, the easy way is to adapt, on the Apple IIgs platform, a game that already exists somewhere else (on another computer, on an arcade machine, on a game console). That seems to be a good compromise because the Design part is already done, as well as the Graphist and Music parts. Of course the next big problem is to get the resources (graphic, music, sound) and to understand the game logic.

The MAIN reason why poeple FAIL to port a game to the Apple IIgs is because they were

unable to get the resource from the original game. The first advice is : DO NOT START TO CODE YOUR GAME IF YOU DO NOT HAVE ALREADY ALL THE GAME RESOURCES. I say ALL RESOURCES. If not, you will end up with unfinished games like :

- Space Harrier
- Morteville Manor
- Defender of the World
- Super Mario Bros
- Bulla Demo
- Mini Prix
- Oil Landers
- Blue Helmet

...

Of course you can easily get the sprites and background of one game level and start to program your engine. But you will quickly face the need to get the other resources and at this time, you will probably give up because getting graphics and music is a boring job. The fun part has been done (programming) and the motivation to complete the game will be gone. Get resource from an existing game is a very complex process involving retro-engineering. 70 % of the complexity of a game port is to get & convert the game data. No game data, no game ! If you think you can re-program a chess game just by looking how the game runs, you are wrong ! Some game have a complex internal logic (Sim City), make sure you have collected all details before coding it.

The second advice would be SOLVE YOUR TECHNICAL ISSUES FIRST. Do not wait to be in the middle of the game to think about how to write a scrolling, display a sprite, play a music... If not, the technical issue will become your first interest (an no more the game itself) and you will switch to another project which to provide a technical solution to a problem. This is the Super Mario Bros syndrome where the programmer has started a game and end up with an animation library (and left the game as is). So prepare first your routines and apply them for the game. Do not try to re-invent the technical part in the middle of the game. Doom at 1 Frame / sec is not a First Personal Shooter, it is (at best) an adventure game ! Make sure your game engine can handle the animation.

The third advice is how to fight against the middle project crisis. You have started a new game for several weeks, you are in the middle of the code, the fun parts have already be done (usually programmed first) and the end is yet far from now. You have less motivation and you start to think about your next game. The next game is always the best one. It seems always more interesting to do. There is always a next project, something that looks more fun. More you start to think about how the next project is cool, more you think about jumping directly on it, letting the current one unfinished. We have all a lot of energy to start new project, but ending something is difficult. To fight against the middle project crisis, the idea to work with a Team, not alone. If you are at least 2 people working on a project, everyone is motivating the other one. Most of the unfinished work are done by single developer.

If you don't want to loose your motivation in project, do NOT make public any pre-release. You may have half dozen of people testing the software but do not make it public. The audience will be happy to receive it the first time, but no one is interesting to receive every week a 0.12.alpha_3 release. You might also loose interest in your own product once the whaou effect of the

announcement is done. Do it well, end it, test it, and give it to the community. Avoid any announcement while the project is not 100% finished.

Olivier