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

Subject: Re: How to FAIL to make a Game on the Apple IIgs
Posted by blondie7575 on Sun, 05 Oct 2014 17:24:33 GMT
View Forum Message <> Reply to Message

This is good advice, which also applies to modern indie platforms such as mobile and Steam. Perhaps even more so, because the fidelity is higher and the market more crowded, so the need for good art and design is even higher.

I think a lot of programmers don't realize that there are LOTS of artists out there who would be happy to work on your game. Many will even do it for free. There are also great free (as in beer) sprite libraries you can use, no strings attached. Much like programmers, artists love what they do and crank out great work even when nobody is paying them. You can get great custom work for a small fee, as well. I've done this for a few of my games. Just go into the indie game forums (tigsource.com or indiegamer.com are good ones) and ask for help. Describe your game, and ask for an artist to make assets. If you can offer a small stiped, like $100, you'll get a ton of responses. You'll even get responses if you can't offer any money, but be prepared for the fact that they have no obligation to see the project through, and may move on at some point.

I will play a bit of Devil's Advocate here and say there are some times when it's okay to get started with no assets. If you have a radical idea for a gameplay mechanic or engine architecture, I think it's worth prototyping that first. If your idea is crap, better to find out as soon as possible.

Something I have done a number of times is to build the whole game with placeholder assets (little boxes or font characters, for example). This does two things for you:

1) Forces you to refine your design. The best games are fun even if the artwork is just colored boxes. Think about the running and jumping in Mario. That has a feel that is separate from the art, and is fun to play even without sprites.

2) It defines the scope of art you need. It's a lot easier to get help from an artist if you can hand them a list of exactly what sprites you need, and in what sizes and formats. If you have to go back and request changes 100 times because you didn't think about how video memory layout was going to affect your collision box alignment or something, you're going to lose their interest in a hurry.

But overall, I agree with Olivier. I'll add that it's easier to maintain your motivation to finish when you can see real art in there. Seeing the game "come together" is huge. It's more difficult to keep going when you're staring at plain boxes for weeks on end. On a few projects, I started down the path of "make it fun with colored boxes", but halfway through went and got some artwork to put in

because it was too depressing without it. Doing that really motivated me again, because it felt like a game all of a sudden, instead of a complicated geometry problem.

Subject: Re: How to FAIL to make a Game on the Apple IIgs
Posted by AppleIIGSMarc on Wed, 08 Oct 2014 17:36:26 GMT
View Forum Message <> Reply to Message

Another suggestion I have is to resist the temptation to get overly ambitious on your first title. Try to start with a relatively simple game and see it all the way through to completion. It will almost certainly take longer than you think and the experience you gain from it will be very helpful when determining the scope of your next project. I can't tell you the number of times I've see a new game developer set out to create the next Ultima or Skyrim without ever having made a game before and then abandoning the project halfway through after realizing just how much work is involved.

Subject: Re: How to FAIL to make a Game on the Apple IIgs
Posted by blondie7575 on Wed, 08 Oct 2014 23:32:15 GMT
View Forum Message <> Reply to Message

Yes, great advice. If you've never written a complete game before, it's very easy to underestimate how much work it is. The last 20% takes 80% of the time. Debugging edge cases, polishing animations, sound, menus, etc. Testing on all the different hardware in various situations, etc. It's surprisingly time consuming.

When starting a new game, I generally set out to make the absolute smallest design I can that achieves the idea. Try to isolate a single gameplay mechanic, and implement that. Think Flappy Bird, not Super Mario Bros. You can always add to it later if you finish and still have motivation. This is also true to the spirit of retrogaming. Most retro action games are so-called "one-mechanic" games, along the lines of modern casual equivalents like Flappy Bird, Flight Control, or Angry Birds.

I also recommend steering away from level-based games. Making maps and levels is an extra pile of work on top of everything else. Make it a one-screen game, if you can. If you must have progression of some sort, consider procedural content such as Roguelikes. Procedural content engines also happen to be fun to write (although still considerably more work than a one-screen game). Level-based or map-based games also usually need more tools (map editors and such). By the time you get all the tools written, you'll have lost interest in making the game.

Subject: Re: How to FAIL to make a Game on the Apple IIgs
Posted by 6502_workshop on Sat, 25 Jun 2016 14:26:46 GMT
View Forum Message <> Reply to Message

The OP Sounds like great advice, and lots of other good comments as well.

I particularly appreciate the points about the mid project crisis and solving the technical challenges first. I made several attempts at writing games in the 1980s and never came close to finishing any because I kept starting over with better techniques. Looking back at it, in reality I was unconsciously going through an extensive exercise of learning Apple II game programing techniques, and didn't know nearly enough to finish anything. But, I was frustrated because I didn't see it that way at the time. A great way to figure our if you are "ready", I think it to follow the advice in the OP about solving the key technical challenges first. Keep doing that until not only they are solved but they are solved in a way that you feel good about, which will reduce the temptation to start over. Then make a game from the toolbox you create.

I'm working on a new Apple II RPG right now (Nox Archaist) in assembly, my first attempt since the 1980s and it is going very well using this approach. The game engine currently has a fully functional tile based map, MOB sprites that randomly generate and attack the player, transport objects like boats and horses with special features, enterable towns with NPCs operating on schedules, etc. Even at this point, I haven't done much work on the storyline, or the detailed gameplay mechanics. Once the game engine is fully build out with the framework of a combat system, dungeons, inventory system, magic system, merchant transition system, etc, then I'll start to design the gameplay mechanics and storyline. This will require a little refining of the engine but I've designed it in a modular nature so it shouldn't be too bad and by taking this approach I hope to avoid coding myself into a corner, so to speak.