

Forum:[Hack Forums](#)

Thread: [Virtual21 - 21 Game Simulator](#)

VirtualFrost - September 23, 2016, 2:57 am

<https://github.com/VirtualFrost/Virtual21>

This is the second project I have completed in C# so all feedback is really appreciated. Also I want to possibly try get it so it saves data so each time you quit and come back it all stays there, just not sure how I want to save it unless I make it create a directory on their computer under My Documents or something.

Apart from that, feel free to try it out, break it etc and I hope you guys like it!

abratar - September 23, 2016, 3:23 am

You could make classes so that not everything is in the main class. And e.g. in "standBtn_Click" you do Random card = new Random(); every time you click that button. That's not a good way of using random. make sure to initialise it only once for the lifetime of the program, so it actually is random. A way would be to make it global.

pbdfigsql - September 23, 2016, 4:19 am

Just a bit of constructive criticism, so please; do not take offense to what I say.

1. Use structures!

- Using structures for even small projects can make a lot of a difference if you wish to further the project. (even if you have no plans to, it's still good practice to block your memory).

2. DO NOT initialize things more times than they need to be!

- While it may not seem like there's any point in conserving little amounts of memory in very small projects, it's crucial to get into these good habits early. (remember that your CPU has to process these and one line DOES NOT equal one instruction for the CPU to process).

3. Cleaning your projects before publishing!

- No one likes a messy file structure. Keep it simple and if using Visual Studio, go to Build->"Clean Solution" before publishing. It just removes the unnecessary clutter and personal information linked to the solution.

4. Documentation!

- You did this for a couple of things but when sharing projects or if working in a team, it is very wise to document every so many lines. (Some people even do every 1-3 lines. These people are good to work with in teams).

That pretty much sums up what I have to say about this!

and as for your question pertaining to the saving of the data; I always save my data in %appdata% under a file name.

(if you wanted to use some of the tips I gave you, what you can do is make a serializable structure and then save it to a file to import/export).

Good Luck, VirtualFrost!

VirtualFrost - September 23, 2016, 6:32 am

(Sep 23, 2016 03:23 AM)abratar Wrote: [\[CITE\]](#) You could make classes so that not everything is in

the main class. And e.g. in "standBtn_Click" you do Random card = new Random(); every time you click that button. That's not a good way of using random. make sure to initialise it only once for the lifetime of the program, so it actually is random. A way would be to make it global. I can't believe I did that mistake again, I did the same thing in my last project and make it so Random initialized everytime, at least I know how to change that, thank you for that.

(Sep 23, 2016 04:19 AM)pbdfigsql Wrote: [CITE](#) Just a bit of constructive criticism, so please; do not take offense to what I say.

1. Use structures!

- Using structures for even small projects can make a lot of a difference if you wish to further the project. (even if you have no plans to, it's still good practice to block your memory). Could you expand on this? What do you mean by use structures?

Quote:2. DO NOT initialize things more times than they need to be!

- While it may not seem like there's any point in conserving little amounts of memory in very small projects, it's crucial to get into these good habits early. (remember that your CPU has to process these and one line DOES NOT equal one instruction for the CPU to process). Yep, fixing that up now after the guy above you pointed it out.

Quote:3. Cleaning your projects before publishing!

- No one likes a messy file structure. Keep it simple and if using Visual Studio, go to Build->"Clean Solution" before publishing. It just removes the unnecessary clutter and personal information linked to the solution. This one is new to me, I'll try that out on my next build

Quote:4. Documentation!

- You did this for a couple of things but when sharing projects or if working in a team, it is very wise to document every so many lines. (Some people even do every 1-3 lines. These people are good to work with in teams). Yeah at one point I remembered trying to document just for practise, just didnt go through with it all the way.

I appreciate the help guys!

pbdfigsql - September 23, 2016, 7:02 am

-- Expanding on my first point.

```
Code int playerBet, playerValue = 0;    //Bet = Players Bet, Value = Players Hand
      int compValue = 0;
      int playerBalance = 100;
      int cardValue;take this block of code for example. It's clunky, unconventional and makes room
for improvement slightly more difficult.
```

How you would format it in a structure it would be like this;

Code //you could make the compValue and cardValue global probably.

```
struct player_s {
    int playerBet;
    int playerValue;
    int playerBalance;
}it now adds a scope.
```

Understanding how memory and optimization might be a little above your skill level (since you're doing C#, no offense) but if you ever decide to learn C/C++ these skills will be of great value to you.

Not to mention it's a dying skill. Most people who program C#/Java have insane amounts of memory leaks and therefore an inefficient program. Take Minecraft for a simple example and compare that to GTAV, which Rockstar only employs those who can properly clean up after them selfs and have experience in the areas where it matters. And look what they accomplished on PS3 with basically next to no resources compared to PC. Rambling now, but I hope that sheds a little more light on the subject. Reading into structures and memory is very much worth your time. Although it doesn't do much good in C#. (sorry if it was a bit rushed)

Good luck!

Zilog80 - September 23, 2016, 12:33 pm

Why structures instead classes?

Very interesting reading: <http://stackoverflow.com/questions/521298/when-to-use-struct>

Discuss! ?

pbdjgsq1 - September 23, 2016, 11:33 pm

(Sep 23, 2016 12:33 PM)Zilog80 Wrote: [\[CITE\]](#) Why structures instead classes?

- 1) Structs are much easier to serialize.
- 2) While it is true it does take more instructions to handle struct's over 16 bytes in actual scenarios, for whatever reason; does not effect performance. I have tested it my self in C and C++, unsure of whether this behavior still happens in C#.
- 3) Optimization works better on structs then classes. This is because 'basic' optimization will find patterns in loops for example, and try to mathematically solve it in maybe 2-3 instructions rather than having to go back to the start with something in the stack being edited.

Obviously it's a fine detail; but at the end of the day really it's a matter of programming style. I use classes when I want to actually reinitialize things a lot.

An example situation:

Code class requests {

//cookies inherits Dictionary<string,string>

private cookies _cookiejar;

public reqres request_result;

public void sendRequest(string URL, reqdat data) {

//web request stuff here..

res = ...; //... is just a stand in place for the result of the web request.

_cookiejar = res.cookiejar;

request_result = res;//obviously this makes _cookies redundant but it's an example.

}//this is a rather crude example which I just came up with out of no where, but you should get the gist of what I would use a class for.

}

Classes are more for methods and storing data with a scope in the code.

While structs are more for "structures". If that makes any sense to you, which it does to me.

Code struct reqres {

public cookies cookiejar;

public string URL;

}

If that sheds a little bit of light and makes more sense to you then good luck in your struct-class endeavors.

Ronan. - September 24, 2016, 4:05 am

To expand on Pbdjgsq's point on structures, it's something you should really get in the habit of doing for any project, games especially.

A great way to teach yourself is to create something like a text-based rpg, this gives you the freedom to implement whatever nifty features you want with the goal in mind that they are extendable. A good example would be having abstract classes for generic weapons and then a derived Sword class or even an inventory with an individual inventory slot class.

It's also worth noting that all members of a struct are public while classes are private. You can use whatever keyword you want and it's usually a good idea to do so anyway, but it's something to keep in mind.

Also look into virtual functions / protected variables as they're something you'll be using frequently when driving from a base class.

And as rightly pointed out, Classes tend to be used when you're implementing functions, structs tend to be for smaller clusters of information you want to store in one structure. That said, you could exclusively use one over the other, I'm not sure how well your development team would receive that though :D

Another great thing to get in the habit of doing is not having everything inside the main function, while for small starter projects it's fine, if you want to start building on this and adding more features, you'll want to segment things more. The joys of OOP is having that as an option, I see so many first years and even second years at my uni do the exact same thing, you'll thank yourself later.

Other than that, great start! If you need any other help feel free to PM me

VirtualFrost - September 24, 2016, 3:43 pm

(Sep 24, 2016 04:05 AM)Ronan. Wrote: [\[CITE\]](#) To expand on Pbdjgsq's point on structures, it's something you should really get in the habit of doing for any project, games especially.

A great way to teach yourself is to create something like a text-based rpg, this gives you the freedom to implement whatever nifty features you want with the goal in mind that they are extendable. A good example would be having abstract classes for generic weapons and then a derived Sword class or even an inventory with an individual inventory slot class.

It's also worth noting that all members of a struct are public while classes are private. You can use whatever keyword you want and it's usually a good idea to do so anyway, but it's something to keep in mind.

Also look into virtual functions / protected variables as they're something you'll be using frequently when driving from a base class.

And as rightly pointed out, Classes tend to be used when you're implementing functions, structs tend to be for smaller clusters of information you want to store in one structure. That said, you could exclusively use one over the other, I'm not sure how well your development team would receive that though :D

Another great thing to get in the habit of doing is not having everything inside the main function, while for small starter projects it's fine, if you want to start building on this and adding more features, you'll want to segment things more. The joys of OOP is having that as an option, I see so many first years and even second years at my uni do the exact same thing, you'll thank yourself later.

Other than that, great start! If you need any other help feel free to PM meReally appreciate the feedback, will try to implement strctures in this and look into the points you've made, especially the main class point as I know that's something I keep doing right now. I appreciate it!

