cook, serve, Delicious®

GameMaker™ Edition

Game Design by

David Galindo

www.cookservedelicious.com

This is not a well coded game.

I made so many mistakes. So, so many. If I could rewrite this game, I would, and that’s exactly what I’m doing with the sequel that I’m currently making.

Whereas a lot of the editable files you’ll be seeing will give you great guidance on how to properly build your game, this is the exact opposite. I hope you can view and learn from what I’ve done so that you don’t make the same mistakes. I’ll point out some of the bigger lessons I’ve learned going forward in this doc, but if there’s one big takeaway from all this, it’s that even if you’re practically the worse coder in the world (me), you can build a really nice game you can be proud of. That’s the beauty of Game Maker. I can’t imagine using anything else.

-dev guy (@chubigans)

Game Code and Basics

This game's code has a long, confusing history. It started first as a GameMaker 8 game, then eventually it was ported to Game Maker Studio. Then it was re-coded into a mobile version with extra features, which were eventually ported back into the PC version. Then the game got yet another overhaul with the Steam edition that added achievements, new features, and lots more.

With all of this moving around, I got very sloppy. I would paste and repaste code everywhere when I should have been moving into a centralized standard. I almost never used arrays or many (if any) repeat statements. There were no scripts except for an unused cloud save code for Linux. I would do a significant amount of code within the draw event, something that's not recommended as that slows down game speed. But really, it was because I just got too comfortable with the idea of coding on the fly, instead of coding for the future.

An example of that is the multitude of objects and sprites in this game. If I needed something done, I would create a new object for it, no matter how small. If I had a new sprite, I would create a new sprite object in GM instead of using subimages. This gradually got more and more difficult to wade through, to the point where it was a huge undertaking to adjust even the smallest of issues.

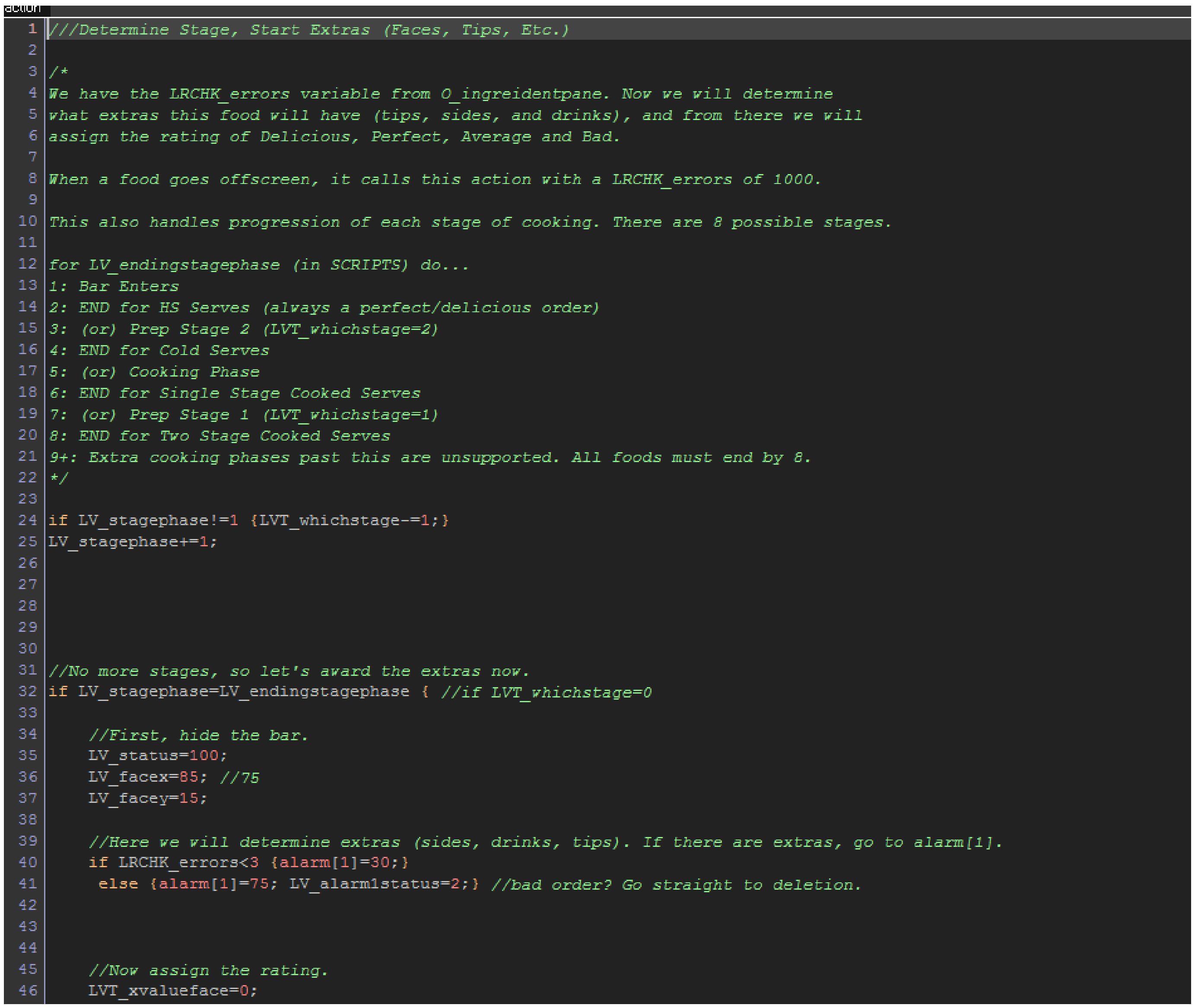
In this doc I want to point out all of the mistakes I made, how I'm coding the sequel, and some of the things I highly recommend doing for your own projects.

Coding on the Fly vs. Coding for the Future

When I first started coding CSD, I never once thought that I would be working on the code after release, aside from the general patches. It never occurred to me that I could be looking at code three years from when I first started, wondering why certain variables were used in certain ways. I either thought I'd remember it (hah!) or thought it wasn't important enough to document. I was basically coding on the fly, and that's a very dangerous way to code.

You'll notice in the CSD code that there's very little documentation within the game. There are comments here and there, but a majority of variables and objects have little to no documentation at all. What is unloadingbacktomainO? You'll have to pour through the code to figure it out (it's the object that unloads resources and then goes back into the main menu). In the sequel I've made it a point to document every object as if I'm handing off the code to someone else to finish, even though I'm likely the only one to ever see this code.

That's what coding for the future is all about: making sure that in a year or even three years' time, the code will still make sense. A lot of the code in CSD 1 makes literally no sense, and that's my fault, not necessarily for the way I coded it but because I didn't bother to document *why* I coded it that way. This can make patching a 2-3 year old game increasingly more difficult.



```
///Determine Stage, Start Extras (Faces, Tips, Etc.)

/*
We have the LRCHK_errors variable from O_ingreidentpane. Now we will determine
what extras this food will have (tips, sides, and drinks), and from there we will
assign the rating of Delicious, Perfect, Average and Bad.

When a food goes offscreen, it calls this action with a LRCHK_errors of 1000.

This also handles progression of each stage of cooking. There are 8 possible stages.

for LV_endingstagephase (in SCRIPTS) do...
1: Bar Enters
2: END for HS Serves (always a perfect/delicious order)
3: (or) Prep Stage 2 (LVT_whichstage=2)
4: END for Cold Serves
5: (or) Cooking Phase
6: END for Single Stage Cooked Serves
7: (or) Prep Stage 1 (LVT_whichstage=1)
8: END for Two Stage Cooked Serves
9+: Extra cooking phases past this are unsupported. All foods must end by 8.
*/

if LV_stagephase!=1 {LVT_whichstage-=1;}
LV_stagephase+=1;






//No more stages, so let's award the extras now.
if LV_stagephase=LV_endingstagephase { //if LVT_whichstage=0

    //First, hide the bar.
    LV_status=100;
    LV_facex=85; //75
    LV_facey=15;

    //Here we will determine extras (sides, drinks, tips). If there are extras, go to alarm[1].
    if LRCHK_errors<3 {alarm[1]=30;}
     else {alarm[1]=75; LV_alarm1status=2;} //bad order? Go straight to deletion.



    //Now assign the rating.
    LVT_xvalueface=0;
```

An example of some of my code in Cook, Serve, Delicious! 2!!

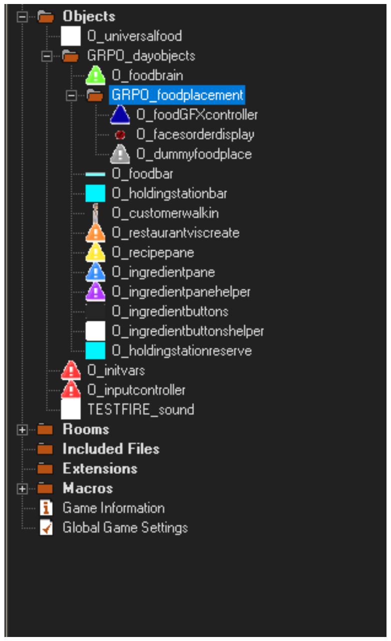
www.cookservedelicious.com

So Many Objects!

I did an incredibly stupid thing. I took something that should have been a single universal object- food creation and controlling it- and turned it into 43 separate objects.

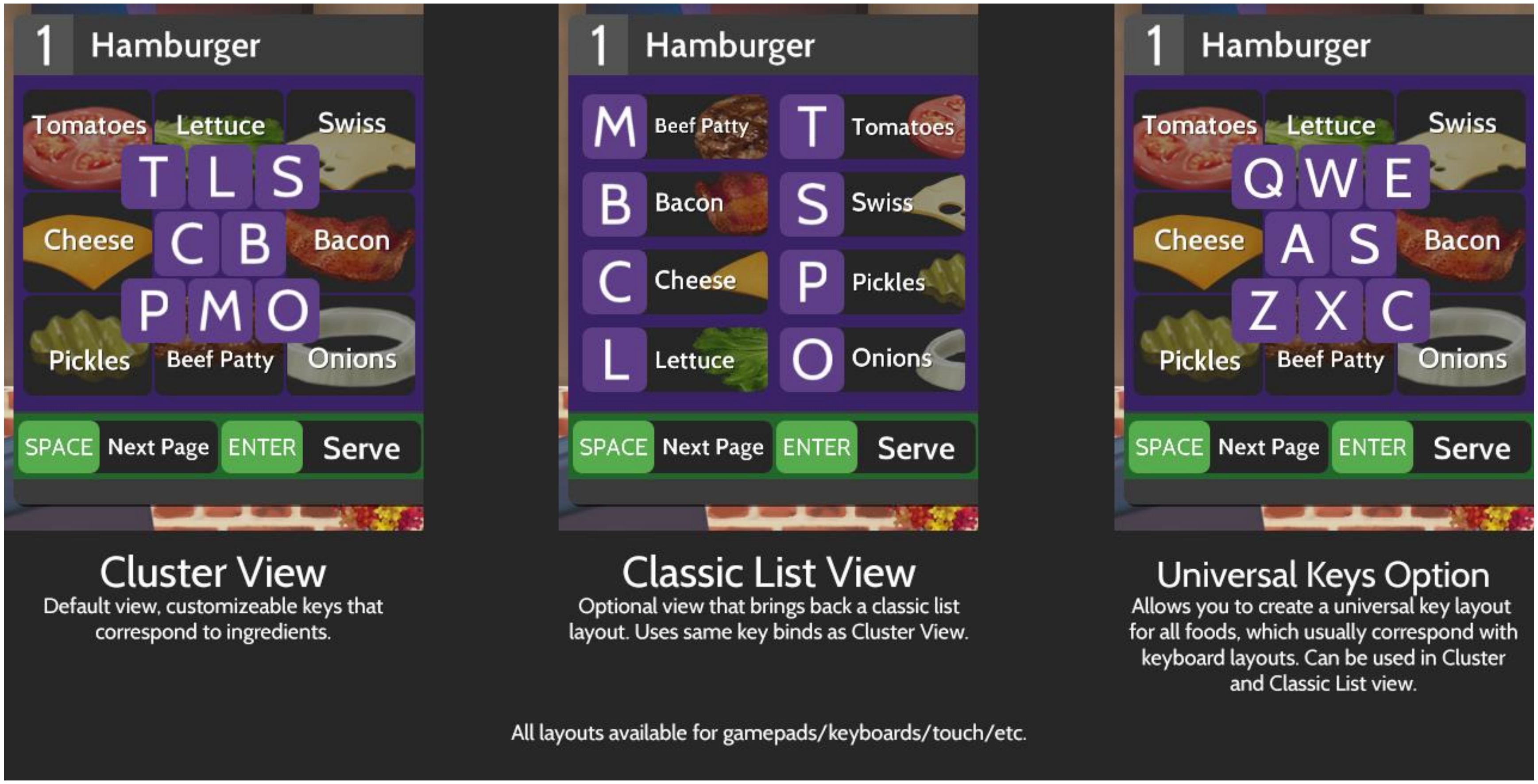
This means that the hamburger has its own object. It has its own menu, it has its own checking parameters for determining whether a food is perfect/average/bad, and it has the same code that I copy and pasted 43 times. If I wanted to move the ingredient menu buttons five pixels to the left, I would have to do it 43 times, and test all 43 objects to make sure they work correctly.

It is an *incredibly* dumb way to handle things. What was I thinking?



In the sequel I made sure to create one universal object for handling all 100+ foods in the game. The benefits were immediate: after getting good feedback on the new ingredients menu for the sequel, I decided to create a new option to display the ingredients list in a list format. In the old game, this would have easily taken a week, maybe longer. In the sequel, I was able to do it all in about 7 hours, and I only had to test it once to make sure it functioned correctly for all foods. Never create more objects than you need, especially if they're redundant objects that share the same code- at the very least, use scripts to handle it so that you're only having to change one line of code instead of multiple. Seems like a no-brainer thing, but man, I sure wish I did that for this game.

The picture on the left shows all the objects in the sequel so far…I might double that by the end of development, but it's a very small object list, with the difference being each object is loaded with code.



Cluster View

Default view, customizeable keys that correspond to ingredients.

Classic List View

Optional view that brings back a classic list layout. Uses same key binds as Cluster View.

Universal Keys Option

Allows you to create a universal key layout for all foods, which usually correspond with keyboard layouts. Can be used in Cluster and Classic List view.

All layouts available for gamepads/keyboards/touch/etc.

The new list menu for Cook, Serve, Delicious! 2!! which I was able to do in hours instead of days thanks to a better code foundation.

www.cookservedelicious.com

So Many Sprites!

Another major issue was creating a new sprite entry in Game Maker for nearly every single image I imported into the game. Not only did this create a massive number of things to keep track of in the Sprites folder but it bloats memory use and slows down compiling of the game. Keep it simple and use subimages whenever you can, as they're much, much easier to manipulate with the subimage variable rather than having to change sprite names all the time. Again, seems like a no-brainer thing to do, but this was something I didn't think was necessary at the time for whatever crazy reason.

Perfect Combo Achieved!
Perfect Day Achieved!
Perfect COMBOS
SUPER
COMBOS Needed
Combos Achieved!
EXTREME DIFFICULTY
STANDARD DIFFICULTY

Memory use is hugely important and by using subimages I would have been able to make the game less prone to crashing on iPad devices, as that only has a limited 512mb on older devices, and the majority of bad reviews came from crashes. I did try to make good use of texture map groups, but because there were so many images, organizing and maintaining those categories across all sprites quickly became cumbersome and difficult.

Scripts, Arrays, and globalvar

Man do I wish I would have used more scripts and arrays. You won’t see much of that in the game, but I highly recommend reading up on some tutorials and introductions to scripts and arrays if you’re unfamiliar with using them, as that would have significantly helped me in the design and coding of the game.

One thing I don’t intend on using again is globalvar, which is being depreciated in favor of global.variable, a much better way of labeling important variables. By using globalvar I had to remember which variables were in fact stated as global, which were local, and which were simply temporary for just that code block. It was a mess of variables and had I used global.variable I would have more easily seen what was what. Most of the code in this game is hard to figure out as you have to look at the initvarsO object to determine what was stated as global.

Final Notes

I have learned so much from building this game that I'm taking into my future projects. My main advice is to be sure to read and re-read the entire Game Maker help file several times, as that has a ton of features that I would have never thought to use had I not read it. Try to learn as much GML as possible and don't rely too much on drag and drop, as that will help you in the future be a better coder. But mainly, code the way that makes you the most comfortable. As embarrassing as some of this code is, you know what? It got the job done. Not in the most efficient way, and certainly not in the *best* way, but there's always the next game to improve on, and that's exactly what I'm doing. Good luck with everything!

www.cookservedelicious.com

COOK, SERVE,
Delicious!
2!!
INTENSE RESTAURANT SIM
coming soon
cookservedelicious.com