How to Fail at Self-Publishing: A Postmortem

Matt

April 24, 2018

Contents

1 Introduction

Making and publishing a game is a very worthwhile and rewarding experience, and one which I absolutely recommend to anybody interested in game-development. This paper will outline my experiences and the things that I have learned from trying to self-publish. Because I worked as a one-person team, this will be focused mainly on people who are working on their own or in a very small team, and while not specifically about Unity - I will reference the engine throughout. This will also focus on self-publishing rather than more conventional publishing methods.

For this project I set myself the task to create and publish a game in a month, from scratch, with no prior experience aside from briefly using the unity engine beforehand. Being honest, the game was not very good and did not do well. As such I **do not** offer advice on the things I did well - but rather a reflection on the things I did poorly, so that perfectly avoidably mistakes can be avoided in future.

For the reader's context: The game I made "Circle Shift" was a 2D puzzler about moving a circle by dragging three points that defined it. I made it in the unity engine with focus on the mobile market. It had customisation options and 40 puzzle levels.

I would recommend reading this whole document through. It is structured in the roughly chronological order of development, but is made to be a lesson about the process as a whole. Knowing where the road ends will make starting it far easier.

I will be touching on the three main parts of the cycle: Pre-Production, Production and Post-Production. I will recommend throughout that you look at guides for each section. I do this because this document is designed to be a reflection on the things that weren't done well, but which we're obvious at the time. As you read though take note that a lot of the things I touch on are things that I realised, far too late, I had ignored. Doing things properly from the start can save a lot of time and effort, and it is my sincere hope that reading this article will help new designers to publish games that are designed properly.

2 Before you begin

How to start the process of making a game that you intend to publish.

2.1 Set Expectations

At the first hour of your development you have a long and winding road ahead and understanding what you will be doing during this time is important. For

context here is a (non-exhaustive) list of the things you will HAVE to do before release:

- Create a road-map
- Design your game
- Prototype
- Play test
- Re-Design
- Program the game
- Create art assets
- Create music assets
- *Create other assets*
- Make a dev-log
- Setup monetising
- Market your game
- Read and comply with legal documents
- Publish your game
- Community management

And this list is by no means definitive, depending on the type of game you want to make, where you want to publish and the resources at your disposal. Do not expect this process to be simple, and understand that a small team will have to wear many hats in getting a game to market. If you don't want to deal with the parts of making a game that involve distribution, or licensing, or art, or marketing then I would recommend trying to join a larger team that has a space for your particular skills.

This is a good time to clear your head of fantasies and understand what will likely be involved during the process. It will be difficult and involve doing a lot of things that you may likely find quite boring. At the end of the process there is a good chance that your game may only be played by a handful of people. Just about all of the games you've ever played have had tens to hundreds of people working on them, even the simplest ones will have large creative teams behind them and often cost thousands of dollars to make. A video game is a big thing to make on your own and you must accept that you're not going to get to make Dark Souls, or Clash of Clans, or World of Warcraft on your first go. Set your sights at publishing *anything*, and build from there.

2.2 Set a Deadline

If you are self-publishing, one of the easiest traps to get sucked into is the "it'll be ready next week" trap. I recommend going into the project with a clear deadline from the start, your time is valuable and not unlimited, and without proper motivation to complete the game there is a good chance it will never get done, or that your time will not be spread properly between your responsibilities.

If this is your first game publishing, I would recommend putting about 100-200 man hours into the project. How long that takes will depend on the size of your team, and your availability to work on the game. If you've only ever done game-jams before this might seem like a very high number, or if you've worked on larger developments it may seem incredibly small.

I put the number in that range for three reasons. The first is that working at minimum wage (for an 18y/o in the UK) for 200 hours would earn £1,180/$1,652 at time of writing. It is very likely that this game will not make a penny of that back. By big games companies standards this is **nothing**, but for an individual it's a pricey spend in opportunity cost.

The second is that, if you're still more of a beginner, you'll likely be many times better at making a game at the end as you are at the start. This is fantastic but means that if you keep working for long enough on a poorly written code-base you're likely to hold yourself up, or worse develop bad habits based on trying to get things to work together. Even worse is the desire to start over from scratch, which while it may be appealing, is not a good way to get to publishing.

The final reason I recommend this range is that it's about enough time to make a great game, allowing yourself less will likely lead to a rushed project, but allowing more may well not be necessary to deliver a fun experience.

A final word of advice on deadlines is to tell people about them. If you're in a team, make it clear what your deadline is and if you're on you're own make sure to include it in a dev-log (more on that later) and tell people around you about it. This creates a pressure that will make it more likely to publish on time.

I would also like to acknowledge that its very likely you'll overshoot your deadline. It happens. Especially when the late stage processes are started too late. Overshooting your deadline is bad but understandable. I'd try and force yourself to avoid overshooting by more than 50% of your original deadline for the reasons stated earlier. Remember, a bad game published is worth more than a good game forever in development.

3 Starting the process

I'm going to lead this section assuming you don't have a particular game in mind. If you do then this works just fine, but consider each section as you go through and how this applies to your idea.

Starting the process is about narrowing down your scope early, entering with too broad a range of ideas or too grand of a design will ultimately lead to problems. Start by removing all the bulk that you can, by picking only what you need and you'll find that far better ideas are generated and actualised than by entering will all the possibilities open but no clear idea of how to make everything work.

3.1 Choose your genre

Picking a genre is not a matter of pigeon-holing your game, but rather understanding the core principals that you'll be building your game around. Of course there is some flexibility and a near infinite sub-classification in each genre, but we're looking for just a broad framework.

Choosing your game's genre should be based around what you want to work on, there's rarely a good game made that the developers weren't interested in making. Along side this, however, you should focus on the skills that your team possess. Certain genres have different key skills associated with them, and this should influence the kind of games you're looking to make.

I can't provide a comprehensive list here, although there are plenty of resources that outline the skill-sets that should be bought to a genre. For example however:

- Point and Click adventures - these games are largely focused on story and art, if you don't have an artist or writer on your team you will find this style of game hard to make.
- Puzzle Games - are very heavy on level design, a good puzzle game only works if there are a good collection of puzzles to accompany them.
- Adventure Games - will require a lot of world building and modelling in order to deliver an interesting experience.
- Simulation Games - are likely to take a lot of complex programming and system management in order to work well and feel interesting.

Although every genre has it's requirements; and all require some degree of art, modelling, writing, music, UX design, and programming as well as the numerous other things that go into a game. There is value in playing to the strengths of your team from the start.

3.2 Pick a platform

At this point you may actually have have an idea of what you want your game to be. This is fantastic but now we need to make one of the critical but difficult decision in the process: Choosing the platform to publish on.

At first, and this is the approach I took, there is a very simple answer. Support all of them. Unity has the option to export to just about every platform going - so why not?

This is the wrong answer. Choosing a platform comes with a number of distinct advantages.

1) It makes coding and designing easier. While a lot of effort is put into cross compatibility, there is just no unified way to make one game work on many platforms. Especially when you take into account shaders, input differences, screen resolution, processing power, and so on. This will save you time and effort down the line.

2) It makes testing easier. Meaning you don't have to build to your phone, to your pc, to a web player. This will keep you focused and more likely to do important tests when they're needed.

3) Monetising. There are different models for monetising a mobile game compared to a PC game.

4) Most importantly, games that are good experiences on one platform are likely not on another. Different platforms have different interactions and this means that the way you design your game should include the platform you're designing for. A controller is a different experience to a mouse and keyboard, which is different to a touchscreen - all offer fantastic options but are separate entities.

To elaborate on point 3; If you're designing for a mouse and keyboard, you have access to a lot of input opportunities this means that games with more complex systems are preferable. Most obviously would be adventure and strategy games. A mouse is also more favourable to precise movement and selection which allows for games that require more accuracy, like when using a hot-bar or old-school sniper simulators.

A touchscreen game offers a different experience. A game built around touchscreen works best when it can be controlled by "natural" movement. These are taps, swipes, pinches and holds. The best mobile games are interacted with using just a handful of unique moves, and some work using a single interaction.

Choosing your medium early **and** understanding what this means for your game will help you develop better ideas and build a better game around the user and how they will interact with your medium.

3.3 Decide what you have to offer

Put simply, you have something to offer. If you've decided to make a game, then you have some perspective, some skill, and some desire to offer to the world. In a small team, you simply won't have the vast array of skills that larger studios use when creating a game. Instead you may be a great 3D modeller, or be fantastic at procedural generation, or just a really great piano player. Play to this strength, and play hard. It's delightful to learn a new skill, and doubtless you will anyway while working on your game, but a learning project is by definition never a finished work. Instead make a game with stunning views of your models, or an endless cave to fly a spaceship though, or a rhythm game where you make people waltz to your music.

Along with this it's worth taking stock of what you don't have to offer, and driving away from that, or understanding that there will have to be compromises made. If you have no artist on your team, then accept that you'll likely be working with shapes as your protagonists. If you don't have a writer then focus on driving your game through mechanics, not plot.

I advise writing down a strengths/weaknesses table and designing your game around that. Ultimately this should help drive you in the direction of a game that really shows the best that your team can deliver.

3.4 Designing the game

Once you've picked your platform, your genre, and have an understanding of what your team can offer, it's time to design the game. There are some key steps in considering how to design a game.

Start with the broadest of frameworks. This is different for each genre but will involve asking 'what is the core experience we want to deliver', and will be the things that most games of the genre have in common with each other.

Then you should think of what your game will do to use this framework, and what it can add on top of it. I do encourage thinking out of the box, the real value that indie games add to the market is not being constrained by traditional ideals of what makes a game - but sticking to the ideals of your genre gives you a box to think outside of. You may choose to base your game around some mechanic, or some story, some piece of music even - but whatever inspires you.

After this think of what will add pizzazz, what will make your game epic, or interesting, or just plain eye-catching. This may seem an odd point, but towards the middle and end of your publishing journey you'll be creating some marketing material, and you'll thank yourself for including these features then.

Finally, scale it back. Odds are you'll create some great ideas and let your imagination run away with you. You're a small team developing in a narrow window of time. You want to be streamlined and effective and this will involve picking the core aspects of the game you want to make, and discarding the fluff. This is hard and will involve trimming a lot of fat but it will be essential. You'll find you get a better response from one idea, well executed, than five with no substance.

Once you've done this it's time to write out your design. This will involve writing down all of the extents of your game, and doing so extensively. This part is ***ESSENTIAL*** and cannot be skipped. Of all of the pre-production stages this will not only save time but also greatly improve the quality of your game.

3.5 Writing out your design

Writing out your design will involve shaping your game and picking the limits on what you will be working on. To do this you should take the design you've been working on and write out the essential components, going into detail about where, and what the extents of any mechanic are. These should be hard limits and are in place to ensure that edge cases are considered. Unfortunately the amount of variables to consider are going to depend on the game, so for this section I will give an example instead.

In this example we are creating a PC game which will take a user submitted photograph and create a jigsaw puzzle out of it, for the player to solve.

When you write your own it should have many more specifications but should include any and all major design choices, any limitations of the program and how they should be handled. It may seem restrictive to insist that maximum file size be 10MB now, but if your programmer can be sure that no image will be larger than that, then they can create solutions in that space. Trying to make everything generic will get out of hand incredibly fast, and while it may seem fantastic to cater to everybody, knowing the answers to these questions going in will save a lot of time and effort.

The most important thing to consider is knowing all your features and their limits from the get go. Every feature that you have to shoehorn in halfway through production is costly and carries the unfortunate burden of potentially

Table 1: Jigsaw Game Specs

File formats that are accepted	.png .jpg .gif
Image aspect ratio limits	1:2 $\Leftrightarrow$ 2:1
If image is larger than allowed aspect	Automatically re-size to fit
Maximum file size	10MB
Minimum number of pieces in jigsaw	4
Maximum number of pieces in jigsaw	1000
Can the user specify how may pieces	No, pick from a set of predefined
Size options	4, 16, 40, 80, 150, 250, 500, 1000
...	...

breaking some other part of the game. I recommend keeping a document outline all of these design choices handy thought production and updating them immediately with any changes.

3.6 Minimal Viable Product / Prototyping

Once you have spent the time to work out what you want to make come a very difficult and incredibly important part. This is the part of the process where you actualise your designs. You should do this part only once you've a good idea of what you want to make but not after you've started any significant work on the project. A minimal viable product is the ABSOLUTE SIMPLEST version of the game that exists. You want to make sure to include the features that you decided make your game important. I advise looking up how to prototype a game but put simply, you want to embody the spirit of your game without investing the effort into the depth of the final product.

There are a few reasons for doing this, and doing it right at the start. The first is to ensure that you, as the designers, understand what is going to be necessary and to prioritise them appropriately. Perhaps you discover that the game is only fun with a solid AI system, or that the atmosphere is stale without a good soundtrack, maybe you'll learn that design of simple shapes is actually perfectly suited to your game without investing in creating complex models. The next, and very importantly, is that actualising the experience as a relative whole from the start will shape the design of the game. This will give you your chance to see what is good, and what is not, and allow you to go back to the start of the design process and re-calibrate what the game is about. This may well involve visiting your write up and changing a lot of the variables. This is your time to experiment with the game and so make sure you're happy with the

experience that you will be delivering.

The third is most important and is why I recommend sharing this version of the game with people outside of your team. Find out if your game is even fun. It's very easy to believe that an idea that seemed great in your head will be great when played but truthfully that just may not be true. There's no shame in this. Discovering early that the game is not fun gives you the opportunity to re-evaluate your choices and will save you weeks of work down the line. Do be discerning with this though. NO game is fun to everybody - but it should be fairly obvious if a core mechanic is lacking or if it's truly interesting. Sometimes this will just mean some tweaking is required and that the core idea is redeemable.

If you don't do this step you will be flying blind, as starting development on the final release immediately is almost invariable a path to failure. Make sure to prototype and test extensively.

3.7 Road mapping

This stage is fairly simple. By now you should know what game you want to make, what your time-frame is, and what you'll need to do in order to make it. What you should do now is combine these into a road-map.

A road map is a structured document detailing what needs to be done and how long you expect it to take. Doing this will help you to adhere to your deadline better, and will also keep your team focused on the task that needs to be done.

Take some time on the document, make sure to allocate time to the strengths that you've observed in your team, because these are the heart of the game. On top of this though make sure that you have set time aside for:

- Creating dev-logs and promotional material.
- Adding monetising to your game
- Setting up publishing. This will likely take a few days to ensure that a good front page has been made. Depending on the road you want to take to publish this may take even longer just to be allowed to publish.
- Testing and Beta-Testing your game.

You're making a small game for a small market but all of these parts need to be considered in your process. Road mapping will be most important when it comes to publishing future games, because it'll help you to create more realistic estimates of what is required in each part.

The key to a good road map is the right level of granularity, try to avoid broad strokes; for example "Make art assets - 1 week". These are not particularly useful as it covers a large amount of work, and with too much flexibility. Usually this will lead to half of the work being over-done and the rest not completed. You should also avoid particularly granular tasks like "Code restart level button function - 2 hours". This is in part due to the amount of time that will be wasted in writing this up. But also because it assumes a level of fore-sight that simply isn't realistic. Generally it's best to set goals with no smaller scope than a half day, and no larger scope than 20% of your project time.

Don't panic if your development doesn't perfectly match your road map. Just make sure that resources aren't being spent poorly, and use it as a tool to recognise when you're falling behind or rushing ahead.

3.8 Start your marketing... NOW!

It's difficult, really sometimes the hardest thing, to promote yourself. But this part of your journey is unquestionably the most important. You may well make the best game ever, with the most passion that all audiences will love and it will mean nothing if nobody plays it.

There's a disjointed misconception that it's pretty easy to fall into that will hit you like a brick wall if you're not ready. That is the belief that your game will get views. It's hard to say, but if you publish a game with no marketing then there's a good chance nobody will play it. There is no magic first 100 views that every game is given and its perfectly possible to never have anybody play your game.

If you don't have a marketing budget then you'll have to be prepared to work hard on getting your views. There's a reason that half of a budget can go on advertising, and that's because it's needed.

The first and almost essential thing to do is start a dev-log. At this point you have a minimal viable product and a road map of how you intend to bring your idea to market. This is the perfect time to start making dev-logs.

If you're planning to publish through a free distributor like itch.io then they have a dev-logging tool I would recommend using. For mobile development creating a blog linked to mobile games would be a good idea. I would also advise making video blogs and publishing to YouTube - this is a medium that will attract attention, while getting you used to publicising your game, and keeping you aware that some graphical fidelity or interest is vital. You absolutely should be able to create a visual experience with your game development and you will be able to attract a small audience doing so.

It will take a while for more passive platforms to draw in an audience, so making them at the beginning is a fantastic way to start building a user base and also to gauge reaction to your game. If people are finding your development boring then that's likely a sign they might not be interested in your game. This isn't a hard and fast rule, people who pay attention to dev-logs are far more likely to be insiders anyway, but it's the best metric you'll have early on.

4 Making your game

You're now ready to make your game. You have a list of your specifications, a road-map for your development, a basic example of your game, and the start of an audience. This section will not be focused on game design, as that's a very broad topic that is better covered in numerous places. Instead I'll mention the important things to consider while making your game from a publishing point of view.

4.1 Keep track of your assets

From the get go you're most likely going to be using external assets. A small team is probably not going to have everything that you'll need to make your game, or at least not the time to produce all assets by yourself.

Now this is where external assets come in. If you're using Unity it's very easy to take advantage of the Asset Store. But this will also include music and sound files, any textures and images, 3D models, scripts, and even fonts.

If you use external assets you MUST make sure to read the licence associated with them. Almost all free assets, and a lot of the paid ones will use an Open Source licence. The specifics will depend on the licence in question, but one important aspect they tend to share is an attribution requirement. You should make sure to document any assets you use in your project and where you got them from.

Failing to do this will result in either a tough attempt to back-track where you acquired each one from or, and this unfortunately is more likely, no attribution at all. Failing to attribute open source resources is unacceptable, bad for the industry, and importantly illegal. In the event that your game gets off the ground, which is the ultimate goal, having improperly licensed products can potentially land you in real trouble. Licences will often apply even if your project doesn't make any money so don't assume that you're exempt from this step even if you won't be monetising.

Keep a running ledger, understand that a free asset doesn't mean that there is no work, and make sure to add a credit section thanking the people who's work contributed to your project by offering their services for free.

4.2 Build in some visually interesting experiences

I have made no attempt to hide how important marketing is, and building in some chances to show off your game as you're designing it will be far more effective than shoehorning them in later. This might involve a high up view of your open world game. Or an endless mode in your bullet hell game. This might mean making your puzzle game have some atmospheric looks to draw in somebody who hasn't gotten invested in the puzzle.

It might seem crappy but on some level you are designing for the people who haven't played the game as much as the people who have. If you can get somebody to look at your page you may have five seconds and three screenshots to encourage them to download your game.

If you're publishing video docs of your progress this should start to happen naturally, but I would recommend taking the time to look at similar games and how they show themselves off to get an idea of the kinds of things you should be considering.

To end this section I'd like to reiterate that you should be updating public dev-logs at least once a week throughout your entire development cycle. Don't fall behind or get disheartened if your audience is growing slowly, just keep going and promoting yourself as much as possible and make sure to actively engage with the people who are showing an interest.

4.3 Test frequently and extensively

There are two kinds of tests you should be running on your game as you make it. Play tests, which focus on the game-play aspects, and Production tests, which focus on the technical aspects of the game. These should be run often, designing without testing is very dangerous and can be the best way to waste time.

Play-testing should be done as often as possible, and ideally after every major system change. Most importantly a play test should be done by people outside of your team. This will mean asking the people around you and anybody that's shown an interest to look at the game. The feedback from these tests are invaluable and I recommend looking into how to play-test properly from other resources.

Production testing should be run by the team, and you'll be testing that everything works as intended. This means all parts of the game function properly. That means that all art assets render properly, all code works as intended, and that there are no holes in any systems. To production test well, you should run your project on your desired platform. Build your game to the platform and

rigorously test any changes. Do not think that something that works in Unity Editor (or your chosen game engine) will work on a mobile, or in a web page, or even as a standalone computer application.

4.4 Focus on what matters

Once you're invested in a project and its intricacies it can be very easy to find yourself obsessing over things which seem important but which don't contribute real value to the game. Your road map should help with this, but I'm mentioning it again in order to push the importance of resource management. If you're going to bring something groundbreaking to your game then it should be the heart and soul of the user's experience, and doing so will add a lot of value to your game. But you don't want to reinvent the wheel for every aspect. It's a good idea to use every tool at your disposal at this stage in your career to make something that works well enough.

My example of this was trying to create an anti-cheating system in my game. This involved a lot of complex encryption and obfuscating just to ensure that somebody who was determined couldn't hack their score. This was unnecessary and a waste of my time. I would suggest asking yourself if putting another hour of work into any part of the project really adds that much value to the final product. In my case I could have been fixing my lighting engine, creating levels, or making a dev-log. All of which would have been more valuable.

I'd also recommend this as a great opportunity to engage with the community. If you're having a problem with a system or an idea, then asking on forums and message boards will do great things in two ways. It will most likely net you a well thought out answer that saves your hours of development time somebody else already spent figuring out the same problem, and will give you an excuse to show off your game at the same time. Be sensible doing this, people will get bored if you bring them problems that could easily be looked up anyway, or if you are using a discussion board as an advertising platform. But played right this lets you make the most of your community.

5 Publishing

This is what every part of the process has been leading up to. Publishing your game is a very important step in any developer's career. Getting to this stage will be difficult and there are some important rules to follow as you reach the end of your project. Starting with the most important:

5.1 PUBLISH

All good things come to and end, and this project will end with you publishing your game. It is a very enticing to leave a game in development forever,

especially if its something you're passionate about. Sharing your work with the world means opening yourself up to judgement and that's not easy. Its likely your game will never feel absolutely finished which makes publishing hard. But that's what we're here to do.

Make sure that everything you set out to do on your road plan is done, and that you're not trying to do anything excessively more. Let your dev-log know that you're planning to publish, this will help drive interest and keep you on track.

Failing to publish means missing out on the opportunity to learn about this part of the game making process. You've been working hard on a game - and at the beginning we calculated the cost you've invested by doing this. You owe it to your team, to your fans (those you have, and those you will make) and to the industry to put yourself out there.

This will almost certainly not be your best work. But it is a place to grow from, and you can always feel proud that you have done something that so many game developers don't do. There is a trope in game dev as there is in literature, about the unpublished author. You will be breaking away from the crowd by putting yourself out there and that will put you head and shoulders above the rest.

And don't forget that digital media is fluid, perhaps in making the game you've discovered there are so many more things you want to add to it. Resist the temptation and publish your first draft. You can always come by and add more later - the BETA status is very common in indie for this exact reason. Don't let ambition hold you back from publishing on time.

5.2 Seriously Publish

I will not stress how important it is to take the plunge enough, but I will offer some practical advice. For each platform there is a very limited number of ways to publish. Publishing for PC is the easiest, I would recommend using itch.io or some other indie distributor if this is your first time publishing. Be aware that larger distribution platforms like Steam have far more restrictive entry standards, and if you are going to pursue this route then you should make sure to do all of your research from the start.

Apple and Android must be published through their app stores (App Store and Play Store respectively). Neither is free, at time of writing an apple publishing licence costs $99/yr and a Play Store account costs $25 for a lifetime account. Make sure that you can afford these costs if you will be developing for these platforms.

Get some experience using these platforms before publishing day. This will ensure that you're ready for the work of publishing in advance. Some things that most platforms have in common however:

1. A name
2. An icon for your game
3. A front-facing image to link to your game
4. 3-8 Screen shots
5. A trailer (30-60s)
6. A short description
7. A long description

Understand your platform, and take a look at the games around you to see what their doing and how it works. Try to avoid looking at high production games - instead focus on your modestly successful peers. They'll give you a good idea of what makes a game appeal.

5.3 Work on your pitch

This is the last step of your marketing. You *can't* afford to short change this section. I'll go over each of the key points with some tips but I insist that you take the time to read real design advice which is plentiful online.

Name Pick a name that is unique, memorable, and simple. This is a hellish task and something that can really take time and effort. If you choose a name that is too similar to more popular apps you'll find yourself at the bottom of the list and difficult to find. There are entire fields of study behind naming a product so don't rush this.

Icons, Images, and Screen Shots Look at any game on your distribution platform. Decide if you like it from a quick glace. Then take your time to actually look at the promotional material. If it's a game you're interested in, odds are there are some interesting images - the art style is on show, there's a demonstration of the core game-play, and it appeals to the aesthetic promise (that is, an action game will seem intense - a puzzler will seem fascinating). Please look at design tips for these but take you time to pick the 3-8 most grabbing frames of your game. They are what will make a person hit download.

Trailer Your trailer is there to intrigue the viewer. If you can watch the trailer without thinking "I want to give that a go myself". You've failed. This should not be the **best** 30 seconds of your game, but rather a teaser to make a person who has no idea how your game works want to get involved. You shouldn't skimp here, and this is a great time to have been making a YouTube log because you will have a good idea which parts of the game people pay the most attention to.

Descriptions Here you walk a line between being informative and being tantalising. Your short description is a chance to drop the buzzwords that your demographic will be interested in. This is also a chance to do something that, until now, we've not wanted - put people off. No matter what goes into your description some people will be interested and some will be put off. You want to ensure that people's expectations are well set coming into your game as this will frame their options when leaving a review or comment - which is what will draw people in. Your longer description is much harder and you should write honestly and informatively. People who read the long description are already very interested in playing the game, so you should not be afraid to spill your passion into this section. As with all sections, read a guide before you write these.

This section is not designed to scare you, but instead to push you to understand what kind of work will go into publishing. When looking at your deadline and road map you should absolutely consider that this will represent a serious number of man-hours. A half-arsed or rushed release page will turn people off of your game, and this can be devastating if you've really put a push into your game. Don't fall at the last hurdle. Be proud of what you've done, nobody could see your game better than you now.

6 Promote

So you've put your game up. Your avid fans have already jumped at it and that's fantastic! But now is a great chance to push this game on as many people as possible.

There are plenty of places to find people who want to play a new indie game. Start with the people in your life, it may seem hard, but if you think of it as a work of literary or artistic work - which it is - then it's no stranger than showing off a great painting or short story. People will appreciate it. When you're working on your next project you may have new play testers and an established fan base.

Then look at the forums where you've posted asking for help, or looking to offer advice. Remember, going in, that you're not here to use the community -

you're a part of it. Let them know about your game and ask for criticism - offer as much as you can in this process, there is a real give and take.

Utilize social media, Facebook, Twitter, Instagram and Youtube will all meet a varied audience. With minimal effort these are the platforms that offer the opportunity to spread the furthest and widest. For the best effect have these accounts set up at the start of your process and update them along with your dev-logs. (With a little work this can be as easy as publishing roughly the same log to all websites).

7 Follow up

It's done. The whole cycle is over and you have successfully published your game! Take some time to rest and look over your project. There are just a few more things to do from now, that will depend on your success and your project.

Follow up with your community. If you've followed all of my steps you likely have a small following built up. Make sure to thank them for their support and make sure to keep them in the loop about your future plans. Making your next project will be a lot easier now you have an established fan-base and the worst thing you can do is to drop radio silence on these people. If you know you'll not be working on anything else for a while, ask if anybody would like to be on a mailing list. If you're going to jump straight back in, leave some teasers and breadcrumbs so that your community feels rewarded for following you.

Game follow up. Your game will fall somewhere on the spectrum of "I never want to look at this again"-to-"Firmly stamped *early access*". No matter where it is, there will be follow up to do. Keep and eye on bug reports and crashes, fixing these will show you're engaged with your players and avoid being slated with bad reviews. Make sure to back up and store your project. If you've been using version control make sure you keep access to your repository. The worst thing is losing your original game files, especially if you find yourself wanting to refer to them later.

If you have stamped your game early access - then you should expect to repeat this entire process again. That will include road maps, deadlines, marketing, documentation etc... Make sure to read up on articles focused on running an early access game especially those about funding such a game for a dramatically extended build schedule.

Autopsy/Post Mortem. These macabre terms refer to reflection on your project as a whole. They are most important if your release hasn't done well but are equally important if it has reached or exceeded expectations. This document is a Post Mortem built around the mistakes that I made while working

on my project. It can take just about any form, a document, a video, a blog post. All that matters is that you make a note of what has been learned from the experience. If all you take away from the experience is having learned more about making and publishing games then you've taken something very valuable.

8 Concluding remarks

I have taken a great deal from my experience, and If I had to distil it all into one remark it would be: "Try it yourself". Really there is no substitute for first hand experience and there will always be things that could be done better. But if you learn and grow from your experiences you will make a great game designer in no time.

I wish anybody who read this the best of luck in their game making.