



THE 7 HABITS OF EFFECTIVE GAME DESIGNERS

This quick guide gives you straightforward design tips. It will help you make your video game more playable.

Why it matters: The amount of information out there on game design can be a lot for those starting to make games.

In the beginning, you just need to make something playable.

This guide cuts through the noise. It gets straight to the point so that you can start designing playable experiences fast.

What this is:

- A starting point for your game design journey

What this is not:

- The “right” or “only” way to design games
- A programming or development tutorial

⚠ **Disclaimer:** You can always break the rules. Once you’ve tried these techniques, experiment with what happens when you break them.

AUTHOR'S NOTE

It doesn't matter if you're trying to improve your game jam skills or are working on a game for millions of players. These design habits will help you make more players happy.

Still, execution matters. Whether you choose to use these habits in your game is up to you. But, after 10+ years in the game industry, they've never steered me wrong.

I hope this guide helps you design great games.



Tell the player what to do.

“What do I do, and how do I win?” is the first thing your players think when they sit down in front of your game.

Make it easy for them to answer those questions.

Why it matters: The goals of a game are what give the player purpose.

If they don't know what to do, they can't progress through the experience. Even open-ended sandbox games like The Sims and Rim World have clear goals if you pay attention.

 **Try it out:**

- Keep your game's goal(s) clear and simple
- Display the goal(s) where the player can see them

- Guide the player through virtual spaces using signposts and visual queues
- Create more than one way to achieve the goal(s) in your game so that the player can choose how to win
- Build your game or feature around one player-goal

Examples:

- “Collect the coins before time runs out”
- “Destroy all the enemies”
- “Talk to the Innkeeper”

👉 **Remember:** If the player doesn't see it, it doesn't exist for them.

When you make awesome features and content, make sure the player knows how to find your hard work. Point them in the right direction with clear goals.



Teach the controls.

There are a lot of buttons on your average controller and keyboard. How will your player know which ones to press?

Why it matters: When someone sits down to play your game, you want them to start enjoying the game as fast as possible.

Your player doesn't want to waste time to figure out the controls. They want to play the game. Make it easy for them.

👤 **Try it out:** There are many ways to teach the controls of your game.

Some methods are better than others, but *any* method is better than no method at all.

- Create a tutorial. This is the hardest method, but if done well, the most effective.
 - Put the player in a sandbox level. Create a safe area in your game where the player can test the controls without challenge.
 - Show the player an image of the controller and keyboard. It should show all the buttons they can use and what those buttons do.
 - **At the very least**, create a text list of your controls. You can throw the list in the description of your game's itch.io page. It can also be a screen somewhere in your game.
-



Give the player variety.

Variety is the key to keeping players engaged in a game. There are simple things you can do to create variety in your experience with little extra work.

Why it matters: Players get bored fast if they see and do the same things over and over again.




Games that make the player ask, “What’s next!?” will keep them invested in the experience.

Subtractive design: You don’t have to do extra work to create variety in your experience. You can subtract elements from your game to create new combinations of gameplay.

For example: Say you have a game where the player has a sword, a shield, and a bow. Instead of creating a brand new element (say, a spell wand), you can remove parts instead.

Make a level that only has:

- A shield
- A bow and shield
- A sword and bow
- Etc...

	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7
	✓	✗	✗	✓	✗	✓	✓
	✗	✓	✗	✓	✓	✗	✓
	✗	✗	✓	✗	✓	✓	✓

Don't add new gameplay to your game until you've maximized your existing game elements. This is how you achieve “depth” in gameplay.


The bottom line: You can get a lot of mileage from a few elements. Extract as much value as you can from your game mechanics and keep the player guessing what happens next.



Know the difference: Interactions vs. Game Mechanics.

Why it matters: Gameplay boils down to player interactions and game mechanics.

But, most game makers don't know the difference between the two.

 **Definition:** A game mechanic is a challenge in an activity that can evolve from easy to difficult.

For example:

- Jumping is an interaction. Not a game mechanic.
- Jumping across a gap is a game mechanic.

Jumping across a gap is a game mechanic because you can make it easier or more difficult.

Here's how:

- If the platform at the other end is moving, this makes the timing more difficult
- If there's an enemy chasing you, this makes it more difficult
- If it's a short gap, this makes the timing more forgiving, which makes the mechanic easier


The bottom line: Game Mechanics regulate how easy or difficult an experience is.

Interactions, unlike mechanics, are verbs that don't have a challenge attached. For example:

- Opening a chest
- Rearranging inventory
- Triggering a conversation

 **Disclaimer:** If you disagree with this definition of what a game mechanic is- that's good.

There's plenty of room for different perspectives. What matters is that you find a definition that works for you and allows you to design your game the way you want.

 **Remember:** Always know how to make your game easier and more difficult. This gives you a lot of control over your design.



Keep the game in a finished state.

There's a way to finish your game on the very first day of development. It's one of the best things you can do to create a smooth development process.

Why it matters: Feedback is the lifeblood of good game design. The sooner your game is playable, the sooner you can get feedback.

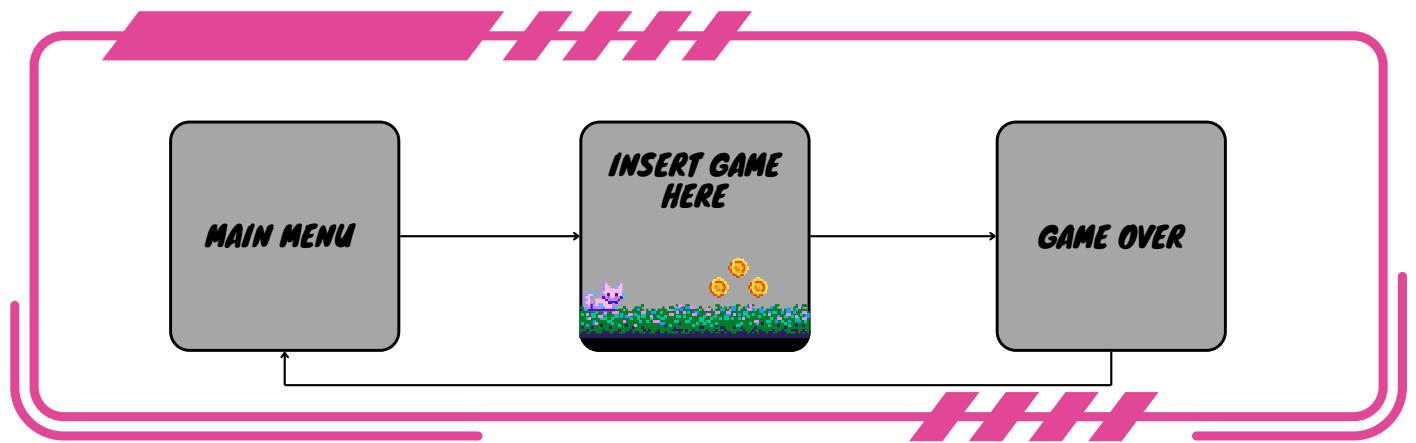
You want to get feedback early and often.

If your game is in a playable state from the very beginning, you can start to collect feedback right away.

That will lead to a better-designed game.

Try it out:

- Begin by creating the start screen for your game. It should show the title of your game and a button that leads to a new game.
- Next, create the end screen. This can be a “game over” screen that says, “Thank you for playing”. If your game doesn't have a game over, you can instead create a quit button that leads you back to the start screen.
- Now, build your game between those two screens. Check often to make sure the flow between your start screen, your game, and the end screen still works.



The bottom line: You now have a game that is, for the most part, in a “finished” state.

You can put the game into someone's hands at any point.

The player can transition from the start screen to the end screen. This is a complete experience that you can begin to receive feedback on.



Make related things similar and familiar.

- If your health bar is green, consider making your potion pickups green
- If an object with a hazard sign explodes, put hazard signs on all other exploding objects
- Make fast enemies small and skinny
- Make powerful enemies big and slow

Why it matters: Great game design is easy to learn but hard to master.

You can make a game easier to learn if you make the elements consistent.

Piggyback on player expectations and existing knowledge.



Doing nothing should be meaningful.

Why it matters: People forget that **not** pushing a button is as important as pushing a button.

Choosing not to do something should have gameplay consequences.

If there's a benefit to doing something in your game, there should also be a benefit to not doing it.

This approach creates tradeoffs. Tradeoffs are one of the most fun decisions a player can make in a game.

For example: Let's say you have a treasure chest in your game that the player can destroy to earn a reward.

What are some ways to reward them for **not** destroying the treasure chest? They could:

- Gift the chest to an NPC to increase friendship with them
- Use the chest to climb onto a ledge that's out of reach

The bottom line: You can always increase the depth of a single interaction.

Think about how the player benefits from not doing the interaction.