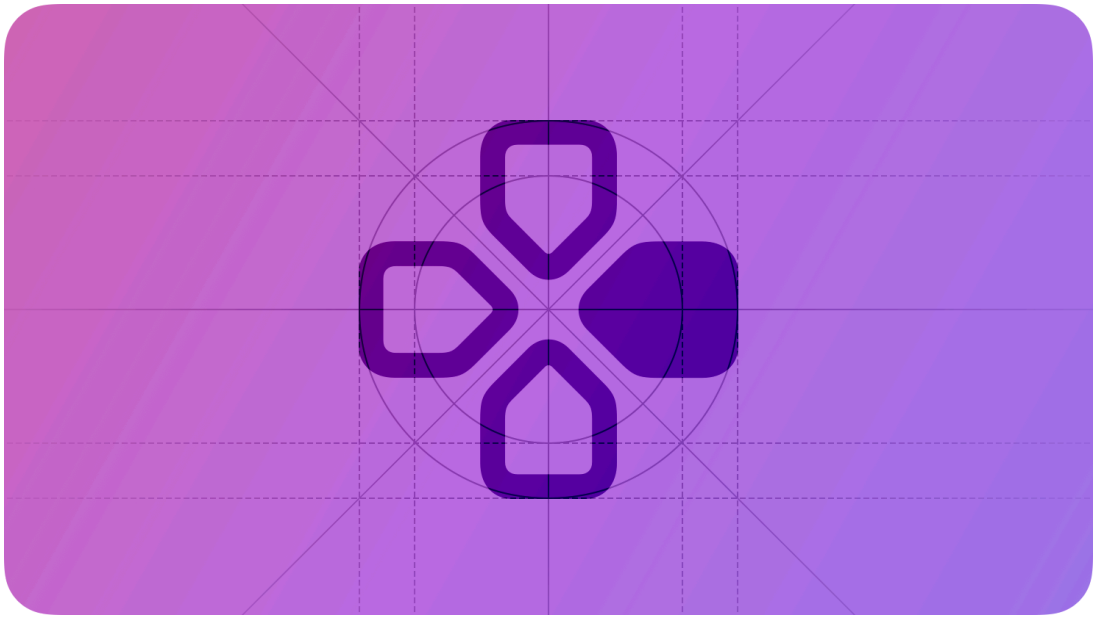
Game controls

Precise, intuitive game controls enhance gameplay and can increase a player's immersion in the game.



On Apple platforms, a game can support input from physical game controllers or default system interactions, like touch, a remote, or a mouse and keyboard. Many players prefer to use physical game controllers, but there are two important reasons to also support a platform's default interaction methods:

- Even though all platforms except watchOS support physical game controllers, not every player can use one.
- Players appreciate games that let them use the platform interaction method they're most familiar with.

For games that run in iOS and iPadOS, supporting touch interaction means that you can render virtual game controls on top of game UI while also letting players interact with game elements by touching them directly.

For guidance on supporting players who use the keyboard to control your game, see Key bindings.

Touch controls

Determine when it makes sense to display virtual controls on top of game views. In general, a game that offers a large number of actions or requires players to control movement needs to display virtual game controls. In contrast, when a virtual control maps directly to an onscreen game element, consider removing the control and letting people interact directly with the element. For example, players often appreciate relocating a game item by dragging it or by tapping its destination, and they expect to open an in-game menu by tapping it. For developer guidance, see Adding touch controls to games that support game controllers in iOS.

Place virtual controls where they're easy to target without letting them obstruct other controls or game content. Consider using larger controls and hit regions for actions players must make quickly or frequently, and smaller ones for other actions. If you need to offer thumbsticks, consider displaying a minimal design that hides when players don't need to use them. As you lay out virtual controls, be sure to support both landscape orientations and take advantage of system-defined safe areas to help you avoid a device's display or interactive features, like the Dynamic Island or a camera housing.

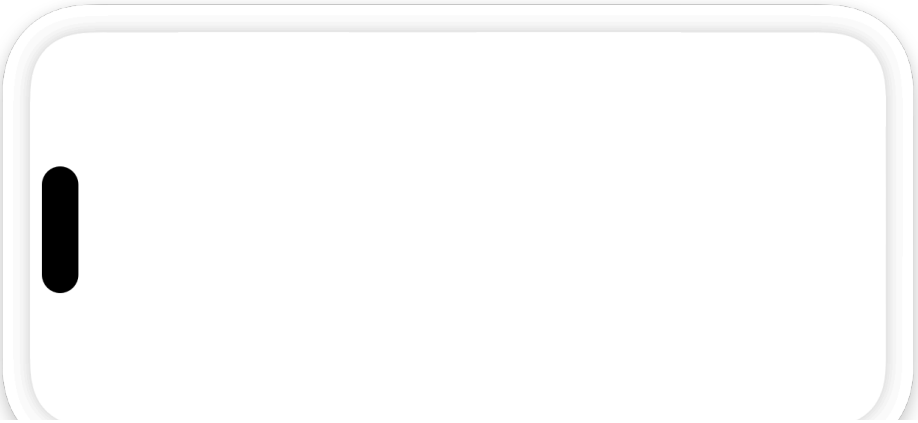
Design for two fingers. People often use their thumbs for virtual controls, so the range of possible actions is more limited than with a physical controller. Consider redesigning game mechanics that require players to press multiple buttons. For interactions that require holding a button, consider making that button a toggle or — when there are multiple actions, such as readying a weapon and aiming — combining the actions into one control.

Placing virtual controls within reach of people's thumbs can make your game more comfortable to play.

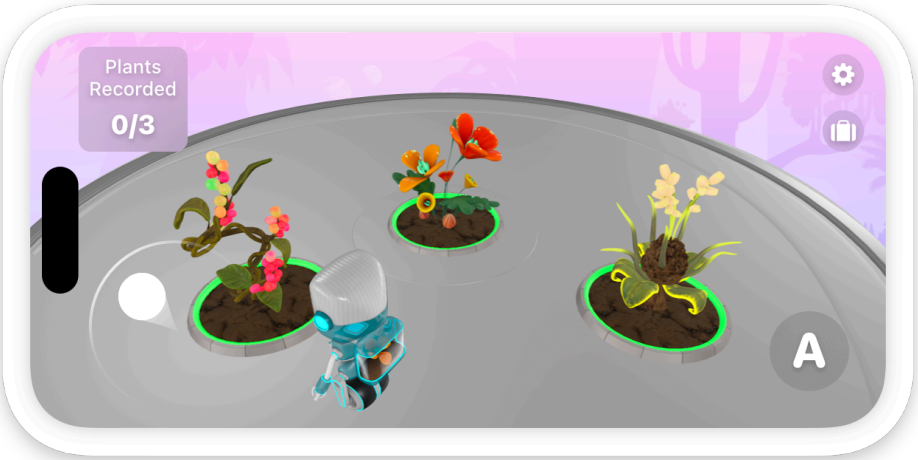
Always include a press state for each virtual control. Without a visual press state, a virtual control can feel unresponsive, making players wonder if it accepted their touch. To provide a press state, you might simply design an appearance that's darker or lighter than the default one, or you might add a glow that appears outside the bounds of the control, helping players see it even when their finger is covering the button.

Consider using haptics to enhance feedback. Subtle haptic feedback can combine with a visual press state to help players confirm that a virtual control is responding to their touch. You might also use subtle haptics to convey the range of an analog control, helping players feel when they reach its limit, or when they engage a different mode. For guidance, see Playing haptics.

Prefer designing expressive virtual controls that communicate the actions they perform. Players can more easily learn and remember what a control does when its appearance visually represents its action. For example, you might design an attack button that displays a graphic of the weapon that's currently equipped, or a movement control that uses an arrow to show the direction of motion.

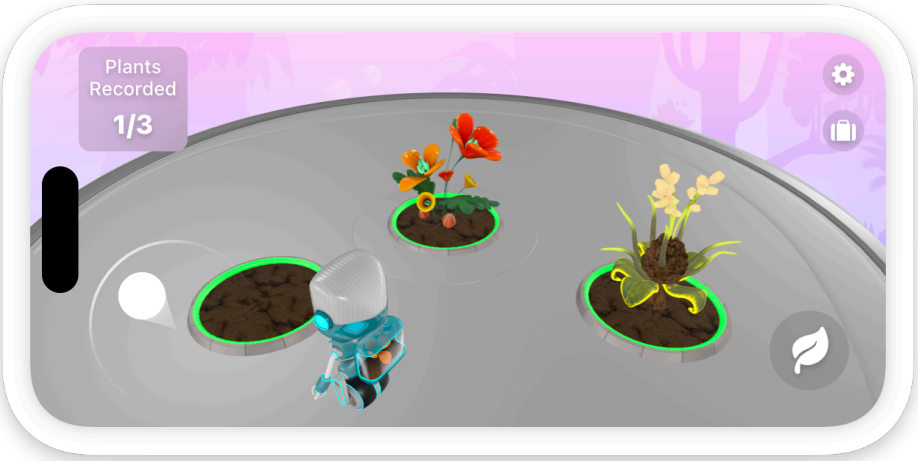


A control that communicates its function during gameplay

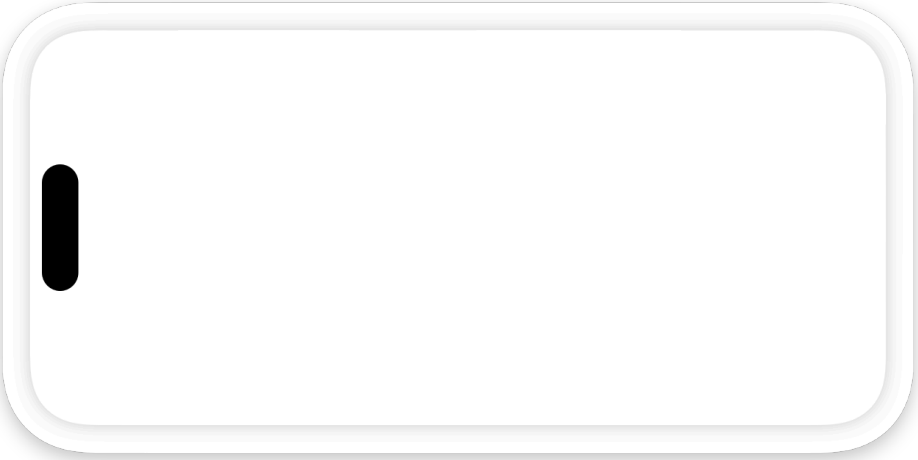


A control that requires players to remember what it does

Aim to design virtual controls that dynamically change to reflect gameplay. Instead of rendering a static overlay that reproduces the buttons on a physical controller, take advantage of touch interaction to give players virtual controls that adapt to gameplay. For example, if the behavior of a virtual control changes based on the current context, its appearance can update, signaling the change to players. Using adaptive virtual controls also means that you can hide a control when it isn't available or relevant, letting you reduce clutter and help players concentrate on what's important.



When the thumbstick moves to the left, the virtual control displays a leaf symbol in its label.



When the thumbstick is at rest, the virtual control hides because it isn't available.

Consider supporting direct touch interactions when it makes sense. Letting players interact directly with elements of your game can simplify control and navigation, and can increase player engagement. For example, players often prefer using direct touch interaction to control a camera, because doing so can result in greater precision than when using a virtual thumbstick. In particular, be sure to let players use touch interactions to control an in-game menu — don't require them to use virtual controls.

Physical controllers

Support the platform's default interaction method. A game controller is an optional purchase, but every iPhone and iPad has a touchscreen, every Mac has a keyboard and a trackpad or mouse, and every Apple TV has a remote. If you support game controllers, consider letting people use the platform's default interaction method to control games, too.

Tell people about game controller requirements. When necessary, you can require the use of a physical game controller. The App Store displays a "Game Controller Required" badge to help people identify such apps and may warn people if they haven't paired a compatible game controller with their device. For developer guidance, see `GCRequiresControllerUserInteraction`.

Confirm required game controller connections. People can open your game any time, even without a connected controller. If your app requires a game controller, check for its presence and gracefully prompt people to connect one if necessary.

Help people understand the advantages of using a game controller with your app. If your game is playable without a game controller, but using one is recommended, look for ways to tell players that using a game controller can make playing easier and more enjoyable.

Avoid requiring players to switch input devices. For example, make sure players can use a physical game controller to navigate all game menus and interact with all essential functions without switching to another interaction method.

Support thumbstick buttons when present. Some controllers include thumbsticks that players can press like buttons as well as rotate. For example, pressing a thumbstick that moves the player's character might change the movement speed, while pressing a thumbstick that controls camera orientation might perform a zoom or crouch. As you consider ways to support thumbstick buttons, be guided by the behaviors that people expect in various game genres.

Customize onscreen content to match the connected game controller. To simplify your game's code, the Game Controller framework assigns standard names to controller elements based on their placement, but the colors and symbols on an actual game controller may differ. Be sure to use the connected controller's labeling scheme when referring to controls or displaying related content in your interface. For developer guidance, see `GCControllerElement`.

Prefer using symbols, not text, to refer to game controller elements. The Game Controller framework makes SF Symbols available for most elements, including the buttons on various brands of game controllers. Using symbols instead of text descriptions can be especially helpful for players who aren't experienced with controllers because it doesn't require them to hunt for a specific button label during gameplay.

Support multiple connected controllers. If there are multiple controllers connected, use labels and glyphs that match the one that the player is actively using. If your game supports multiplayer, use the appropriate labels and symbols when referring to a specific player's controller. If you need to refer to buttons on multiple controllers, consider listing them together.

Supporting a game controller in a nongame app

People sometimes use a game controller to navigate the system and use nongame apps. In this scenario, support each platform's expected behaviors for various game controller elements.

Button	Expected behavior in iOS and iPadOS	Expected behavior in macOS	Expected behavior in tvOS
D-pad	Moves focus	Moves focus	Moves focus
A	Activates a control or selects an item	Activates a control or selects an item	Activates a control or selects an item
B	Cancels an action or returns to previous screen	Cancels an action or returns to previous screen	Returns to previous screen or exits to Apple TV Home Screen from an app's root-level screen
X	—	—	Starts or pauses media playback
Y	—	—	—
Left shoulder/trigger	Navigates left or moves focus	Navigates left or moves focus	Navigates left or moves focus
Right shoulder/trigger	Navigates right or moves focus	Navigates right or moves focus	Navigates right or moves focus
Left thumbstick	Navigates or moves focus	Navigates or moves focus	Navigates or moves focus
Right thumbstick	—	—	—

A controller with a single auxiliary button needs to support the following behaviors:

Interaction with auxiliary button	Expected behavior in iOS and iPadOS	Expected behavior in macOS	Expected behavior in tvOS
Press	—	—	Returns to previous screen
Long press	—	—	Exits to Apple TV Home Screen
Double press	—	—	Reveals multitasking user interface

A controller with multiple auxiliary buttons includes a logo button in addition to the Menu button. Such a controller needs to support the following behaviors when people interact with the logo and Menu buttons:

Button	Interaction	Expected behavior in iOS and iPadOS	Expected behavior in macOS	Expected behavior in tvOS
Logo	Press	Reveals Game Center (if available)	Reveals Game Center (if available)	Reveals Game Center (if available)
	Long press	Reveals Games folder in App Library	Reveals Games folder in Launchpad	Exits to Apple TV Home Screen
	Double press	—	—	Reveals multitasking user interface

Button	Interaction	Expected behavior in iOS and iPadOS	Expected behavior in macOS	Expected behavior in tvOS
Menu	Press	—	—	Reveals Control Center or opens app settings

Platform considerations

No additional considerations for iOS, iPadOS, macOS, tvOS, or visionOS. Not supported in watchOS.

Resources

Related

Feedback

Developer documentation

Game Controller

Videos

Explore game input in visionOS

Supporting New Game Controllers

Change log

Date	Changes
June 10, 2024	Added guidance for supporting touch controls and changed title from Game controllers.