



Color Block

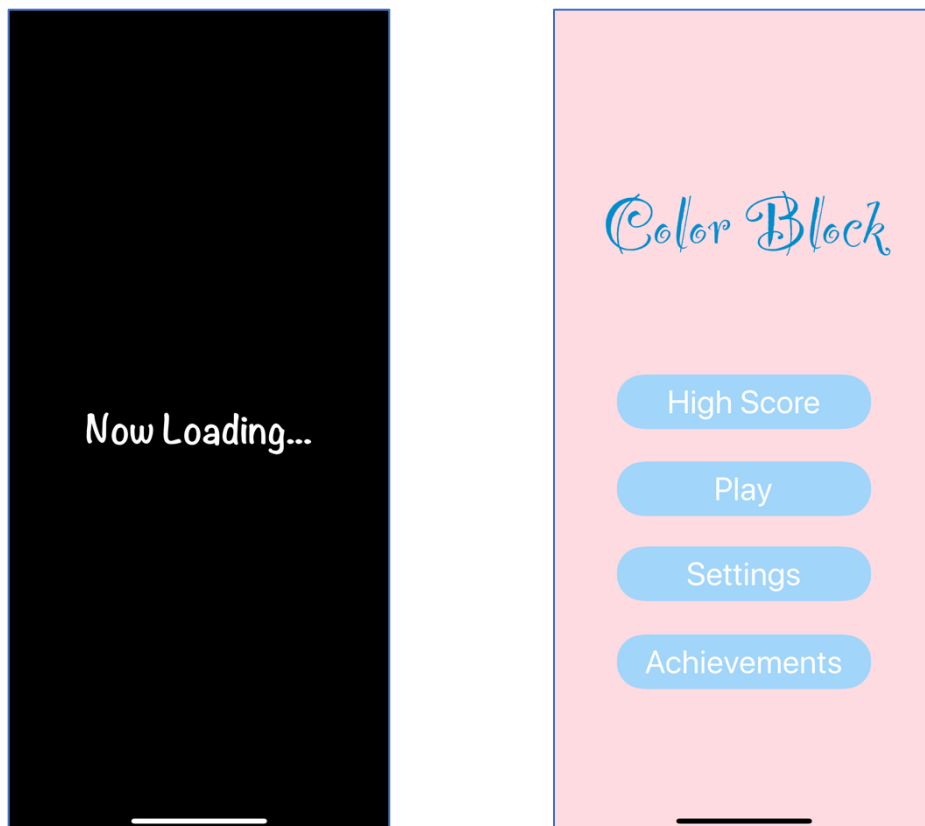
Lin He

Yuning Zhou

User Manual

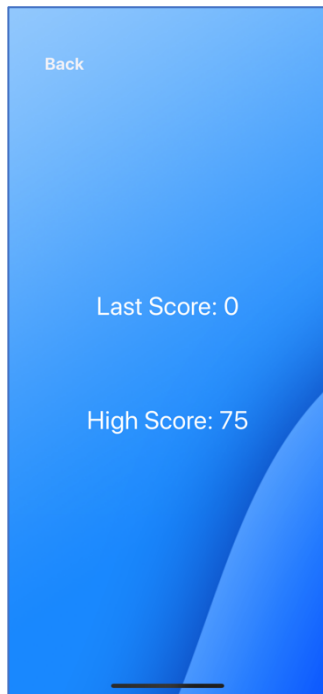
Color Block is a Tetris-inspired block-matching iOS game. It is a casual game designed to pass time. In this User Manual, readers can find instructions of the gameplay, as well as a breakdown of major screens of the app.

Loading & Main Screen



The game loads with the Loading screen. Once on the Main screen, players can find four options: High Score, Play, Settings, and Achievements. Each option will direct the player to a new screen detailed in their respective section below.

High Score Screen



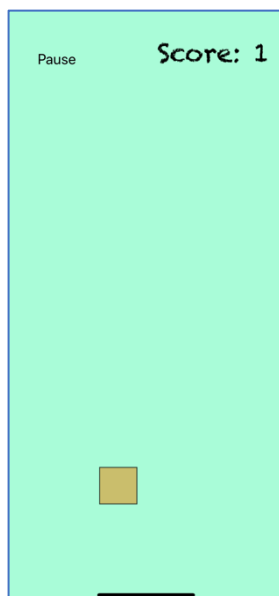
On this screen, players can find two scores:

- Last Score
- High Score

Last Score is the score of the last game the player played. For first-time players, this score will be 0.




High Score is the overall high score across all games. Should the score of the last game be higher than the high score, it will replace the high score.

Play & Pause Screen



The main gameplay screen is as presented on the left. The gameplay mechanism includes the following:

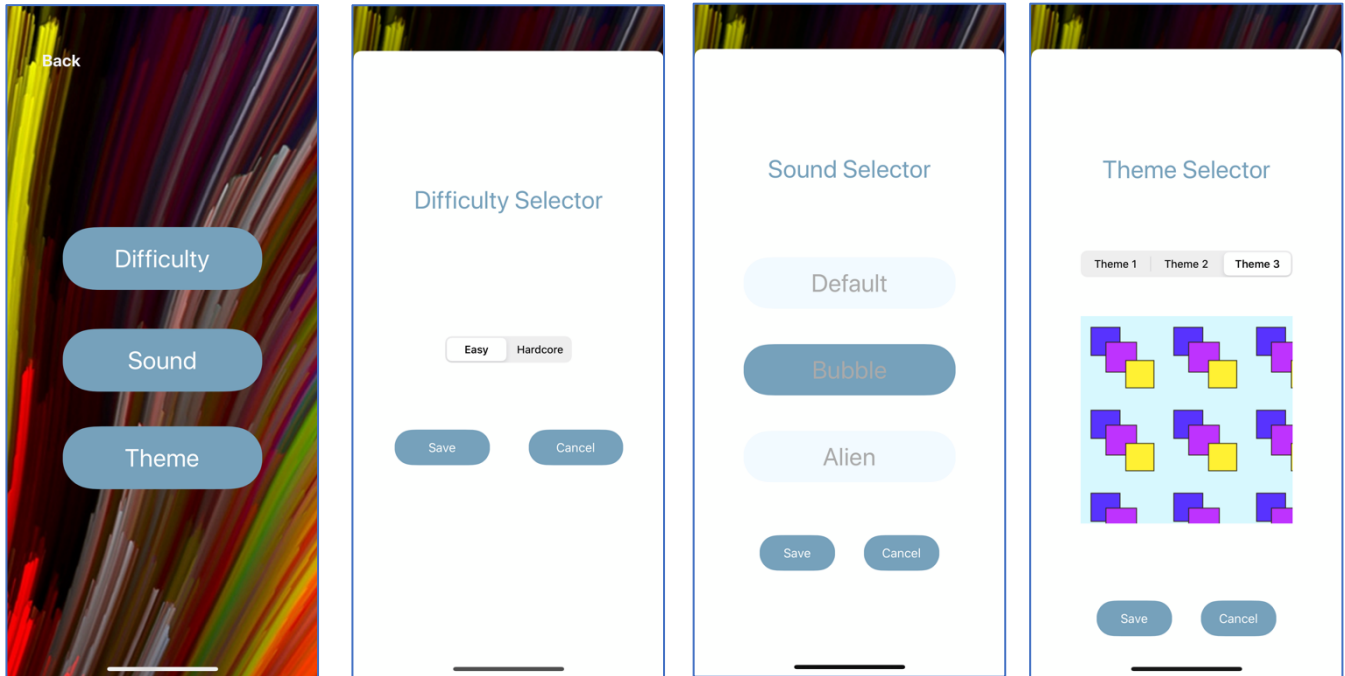
- Blocks of three colors will be randomly generated and dropped from the top of the screen.
- Players can control the block's falling position by swiping the screen.

- If three blocks of the same color stack on top of each other, they are matched and removed.
- If a horizontal line of six same-colored blocks is formed, the entire line is removed.
- Three special blocks will be randomly supplied to help the player in the gameplay:
 - A bomb block 
 - A horizontal rainbow block 
 - A vertical rainbow block 
- The bomb block checks its surrounding blocks and removes all of them. In the optimal case, it removes seven blocks: upper-left, left, bottom-left, bottom, upper-right, right, bottom-right.
- The horizontal rainbow block removes the entire row the block falls into, regardless of color.
- The vertical rainbow block removes the entire column of the block, regardless of color.
- The score keeps track of the number of blocks that are removed.

If the player wishes to pause the game, hit the “Pause” button on the upper left corner of the Play screen. On the Pause screen, players have the option to resume the game, or to quit the game and return to the Main screen.

Settings Screen

This is the screen for players to customize their game, and choose from a variety of options to modify their game:

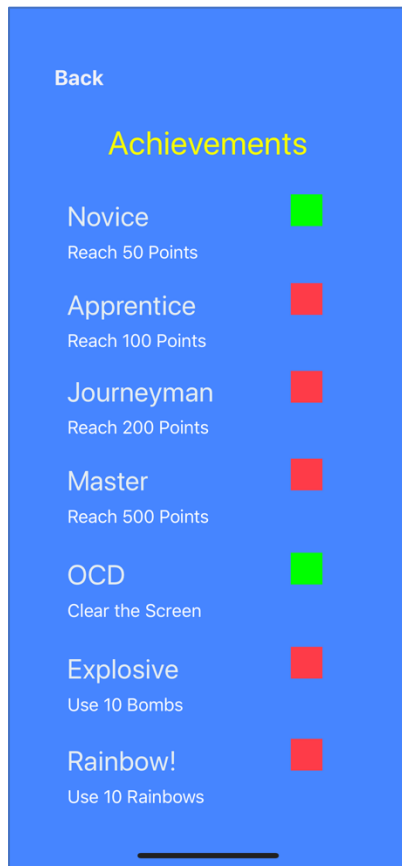


Players have three customizable options: Difficulty, Sound, and Theme. The Difficulty Selector offers an Easy mode and a Hardcore mode, where the game speed and the rate of special blocks are affected. The Sound Selector gives three sample sounds for the players to pick. The sound will be played as the blocks fall down to the bottom of the screen. The Theme Selector offers three color schemes for the gameplay. Color themes include the background, and the three block styles.

Changes made must be saved by hitting the “Save” button. To discard all changes, hit the “Cancel” button.

Achievements Screen

On this screen, players have the chance to check a list of achievements. These achievements include score rewards and gameplay checks.



For demonstration, the “Novice” achievement is reached when the player accumulates 50 points in a single gameplay for the first time. The “OCD” achievement on the other hand, is rewarded to the player when they clear the whole screen for the first time.

The list of achievements are set initially to be red/unchecked. Once an achievement is met, the game will recognize that and check the list by marking the square green. On the example to the left, the “Novice” and the “OCD” items have been achieved.

Technical Documentation

This section provides a logical overview of the code. Developers can refer to this section for a detailed explanation of the major functions of the source code.

Color Block uses SpriteKit as its main technology. Blocks take advantage of SpriteKit’s physics engine to process falling actions and collision detection. Specifically, every block is a physicsBody that experiences gravity within the engine. The falling action is the result of gravity pulling blocks down. The mechanism behind collision detection will be addressed in the spawnBlocks() section of this Technical Documentation.

colorSchemes

This enum stores preset color schemes that blocks and the background can access to customize their color. It offers 3 options, corresponding to the Color Selector in the Settings screen.

didMove()

Similar to the `viewDidLoad()` function in a `viewController`, this function is the first to be called when the `GameScene` loads. This function calls the helper functions `setPhysics()` and `layoutScene()`. In addition, it loads the swiping actions with connections to the two functions in charge of swiping.

setPhysics()

This function sets the physical rules for the engine. It sets the gravity factor that affects block falling speed according to the difficulty setting.

layoutScene()

This function sets the background scene for the Play screen. Elements including background color, score label, and sound are loaded in this function in preparation of the start of the game. At the end of the function, it calls `spawnBlocks()` to start the game.

spawnBlocks()

This function creates new blocks on top of the screen. It uses a random `Int` generator to decide the block's color. Additionally, some values will turn the block into a special block. Depending on the difficulty, the rate of spawning for special blocks are adjusted. The next part of the function sets

the physics properties of these blocks. The blocks and the frame each have their own physics body. The friction, restitution, and rotation of the blocks are turned off to prevent the blocks from moving around and bouncing off of each other. The blocks and the frame are each assign a categoryBitMask to separation. This setting allows subsequent SKPhysicsContactDelegate to detect collision, and take further actions accordingly. Lastly, the addChild() function adds the block to the scene, making it appear on the screen.

SKPhysicsContactDelegate

This extension assigns the physics delegate to the GameScene itself. The main logic of collision processing happens in this section and the sub-functions within.

didBegin()

This function is called when a collision is detected. Since blocks are modeled as circles, side-to-side collision will also be detected. Therefore, this function checks that the collision contact point is on the bottom by calculating the distance of the contact point to the midpoint on the bottom side. When a bottom collision is made, the block is pinned to prevent it from further moving. The function then calls a series of check functions to see if blocks can be removed. Finally, it checks if the game is over, and calls either gameOver() or spawnBlocks() accordingly to continue.

checkVertical()

This function checks if a block matches color with the two blocks below it. If a matching condition is detected, this function will remove the three blocks from the screen. Additionally, it check for the Vertical Rainbow block and process it accordingly.

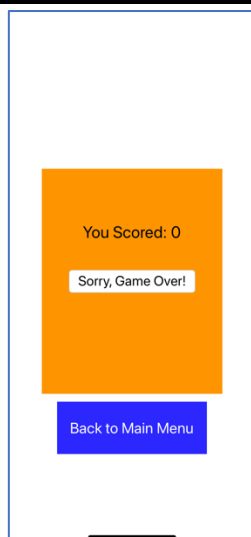
checkHorizontal()

This function checks the horizontal peers of the fallen block. The entire frame is divided by a matrix of 6 x 14. Each of the six array in the matrix corresponds to a column on the screen. This matrix stores all blocks on the screen for processing and calculation purposes. A horizontal check is more complicated than a vertical check, as blocks might have other blocks stacked over them. If a block is buried under other blocks, its removal will lead to the blocks on top falling down. To achieve this effect, a timer is used in the `didBegin()` function. Whenever said condition occurs, the blocks that need to fall down are unpinned, given some time to drop, and pinned again to secure their positions. Additionally, this function checks for the Horizontal Rainbow block and processes it accordingly.

boom()

This function processes the bomb special block. It is essentially a combination of `checkVertical()` and `checkHorizontal()`.

gameOver()



When the blocks stack to a set number (13 for normal mode, 5 for testing mode), the game is over. This function first checks if the game score is over the high score, and updates it accordingly. After all calculations, the scene transitions into the Game Over screen. On the screen, the last gameplay score is displayed, along with a button back to the Main screen.

isEqual()

This helper function compares two objects of any type. It is used by the check functions to compare the color of different blocks.

pauseGame() & resumeGame()

These two functions help pause and resume the game as the “Pause” and “Resume” buttons are pressed.

achievement()

Lastly, this function is called every time the score is updated. It checks the conditions for each achievement, and updates the Achievement screen.

Overall, data is transmitted across different screens. To solve this problem, relevant data is stored in the UserDefaults.standard global variable. Additionally, audio files are played when blocks fall to the bottom, blocks are removed, special blocks are produced, and new achievements are made. Some buttons also use UIBezierPath to soften the edges and improve the appearance.

References and External Code Snippets

The following websites helped in creating this app, including tutorial, inspirations, resources, and debugging information:

<https://stackoverflow.com/questions/39783608/how-to-add-a-swipe-gesture-to-a-node-in-spritekit>
<https://www.raywenderlich.com/71-spritekit-tutorial-for-beginners>
<https://stackoverflow.com/questions/39421856/how-to-prevent-an-object-from-leaving-screen-in-swift>
<https://stackoverflow.com/questions/23177005/spritekit-prevent-sprite-from-leaving-screen-without-bouncing>
<https://stackoverflow.com/questions/29086237/how-to-stop-a-sprite-from-bouncing-off-a-specific-surface>
<https://learnappmaking.com/random-numbers-swift/#random-numbers>
<https://stackoverflow.com/questions/34157852/does-the-useprecisecollisiondetection-property-work-when-the-physics-body-does-n>
<https://stackoverflow.com/questions/33892462/how-to-create-arraylist-of-object-in-swift>
<https://developer.apple.com/documentation/foundation/nsmutabledictionary>
<https://stackoverflow.com/questions/27830093/spritekit-determine-side-a-square-collided-with>
<https://stackoverflow.com/questions/34778950/how-to-compare-any-value-types>
<https://stackoverflow.com/questions/35381709/addressing-a-multi-dimensional-array-of-objects-in-swift>
<https://stackoverflow.com/questions/26686948/how-to-check-object-is-nil-or-not-in-swift>
<https://developer.apple.com/documentation/swift/array/1688398-isempty>
<https://stackoverflow.com/questions/35411306/swift-2d-array-initialisation-with-objects>
<http://spritekitlessons.com/child-basics-in-sprite-kit-adding-removing-finding/>
<https://www.zerotoappstore.com/create-a-delay-or-wait-in-swift.html>
<https://github.com/paxer/ColorSwitch/blob/master/ColorSwitch/Scenes/GameScene.swift>
https://www.youtube.com/watch?time_continue=1019&v=467Doas5J6I&feature=emb_logo
<https://www.pixilart.com/draw#>
<https://riptutorial.com/sprite-kit/example/29867/multiple-uiviewcontroller-in-a-game--how-to-jump-from-the-scene-to-a-viewcontroller>
<https://stackoverflow.com/questions/38149965/sklabelnode-not-showing-up>
<https://www.hackingwithswift.com/example-code/games/how-to-write-text-using-sklabelnode>
<https://stackoverflow.com/questions/25678564/swift-how-to-keep-track-of-and-communicate-scores-between-scenes-in-spritekit>
<https://www.hackingwithswift.com/example-code/language/how-to-convert-a-string-to-an-int>
<https://stackoverflow.com/questions/52373071/userdefaults-standard-objectkey-always-nil>
<https://www.ioscreator.com/tutorials/segmented-control-ios-tutorial>
<https://stackoverflow.com/questions/37738149/swift-segmented-control-default-state>
<https://stackoverflow.com/questions/26261554/allowrotation-in-scenekit?lq=1>
<https://stackoverflow.com/questions/36358609/changing-a-property-of-an-object-from-gameviewcontroller-swift-in-gamescene-swif>
<https://stackoverflow.com/questions/47489849/call-a-function-of-gamescene-in-gameviewcontroller>
<https://freesound.org/people/jeckkech/sounds/391658/>
<https://freesound.org/people/mlrk0/sounds/23215/>
<https://freesound.org/people/plasterbrain/sounds/464909/>
<https://freesound.org/people/JohanDeecke/sounds/369528/>
<https://medium.com/better-programming/playing-music-and-sound-effects-in-a-spritekit-game-e3bdf8cc45e8>
<https://freesound.org/people/Robinhood76/sounds/342131/>
<https://freesound.org/people/CGEffex/sounds/89534/>
<https://stackoverflow.com/questions/19739596/small-delay-when-playing-a-sound-for-the-first-time-with-spritekit>
https://freesound.org/people/Leszek_Szary/sounds/146733/
<https://stackoverflow.com/questions/40470145/errors-thrown-from-here-are-not-handled>
<https://stackoverflow.com/questions/32036146/how-to-play-a-sound-using-swift>
<https://freesound.org/people/elmasmalo1/sounds/377018/>
<https://stackoverflow.com/questions/39524148/xcode-error-code-signing-is-required-for-product-type-application-in-sdk-ios>
<https://freesound.org/people/qubodup/sounds/195486/>
https://www.flaticon.com/free-icon/double-arrow_143238
https://favpng.com/png_download/VHJ1DeJ4
<https://htmlcolorcodes.com/color-picker/>
<https://stackoverflow.com/questions/27049937/how-to-set-a-background-image-in-xcode-using-swift>
<https://stackoverflow.com/questions/27049937/how-to-set-a-background-image-in-xcode-using-swift>
<https://www.color-hex.com/color-palette/2639>
<https://www.pinterest.com/pin/548805904595212710/>
<https://www.ilikewallpaper.net/iphone-x-wallpaper/digital-abstract-line-color-rainbow-pattern-background/36149>

Additionally, code snippets from external sources are marked in the code comments.

Contribution

Lin He – Graphics, User Interface, button creation code, color schemes, research

Yuning Zhou – Main logic, block physics code, main functions, SpriteKit research

Compilation & Note to Developers

This app is written using Xcode. The UI elements are constrained to the screen, so users can install the running app on most mobile devices. This app only runs on iOS phones, and does not support iPadOS.

For testing, a Boolean variable named “testMode” is provided. In the setPhysics() function, users can set the testMode to “true” to enter test mode. Under test mode, the game will terminate when 5 blocks are stacked instead of 13 under normal mode.