Acknowledgements

There is not any human that he learned everything he knows alone. By no means I am not an exception. I want to thank all of my professors and teachers who thought me, specially my C++ professor Mr. kalantar. I learned from many websites that they are the result of charity works, so I want to thank founders of wikipedia and many other websites that I can not remember.
In everyone's life his family and friends have a great impact on him and his situation. I learned much from my family and friends and they have a big share of this nothing that I have. Always they are many things which directly or indirectly affect you; I want to thank them too. As I read somewhere, last but not least I want to thank God who always helped me and never forget me. We always forget him but he never forget us. surely without his help I could not do anything, I should thank him for all of the things that I have, for finding unity (I don't remember how I found this great piece of life) for having a good family and a good friend. For all of the happenings that seemed random to me but they were his works and for everything that is available.

Introduction

Some people think that Screen savers are something from the past. They are right because they are not needed now and people just use them for more visual fidelity. Some people love them and some people disable screen savers. You bought this package so you either love them or need to create one for a customer. In CRT monitors era screen savers were required because monitors would damage if they display a constant image for long times. Oh there is another possibility there; you just want to know what they are and how you can create one.

How can I create my own screen saver?

It's easy.
1 import the Screensaver Builder package into your project. Don't change its folder structure.
2 create 2 scenes. A scene for the main screen saver and another for settings
3 add an instance of the ExitController prefab from Screensaver Builder folder to your main screen saver scene (you can use scenes in screen saver builder folder).
4 set its properties

settingsScene: name of the settings scene that the screen saver should load. This property is not important if you don't have any settings scene.
waitingTime: this property will cause the exiting events to don't accurse before a given time. Think you have a splash screen that you want to show before allowing the user to exit the screen saver with mouse/keyboard events. This is not a good thing but I put it here for your flexibility.
mouseMoveMinDelta: this property is the minimum amount of mouse movement between two frames to be considered as mouse move event. You should set it to something larger than 5 pixels to avoid exiting with random mouse moves. (5 is a value that I told randomly, test it to see what is the best value for the resolution that you use)
exitWithMouseMove: if true then screen saver exits if the user moves the mouse.
exitWithMouseButton: if true exits the screen saver if the user presses any mouse button
exitWithKeyboard: if true exits the screen saver if user presses any key.
exitWithEscape: if exitWithKeyboard is false you can check this to allow the user to exit the screen saver with ESC key.
5 disable the display resolution setting in player settings. If you don't disable it then unity will show the screen resolution dialog before running the screen saver. Don't forget to choose a resolution for your standalone. Don't choose a too high resolution because some of your users might have low end graphics cards. You can make it customizable in your settings but in this way the screen resolution will change 2 times because first unity changes it to the default value in build settings and then you will read it in your script from PlayerPrefs or any other place and change it again.
6 don't forget to write the screen saver itself.
7 build a windows standalone from your application with your main scene and settings scene. The main scene should be the first scene.
8 from the Tools menu choose Build Screen Saver. Answer the question that if your screen saver has any settings scene or not. If you answer yes then it will load your settings scene when the related button is pressed but if you press no, the starter shows a message box that tells "this screen saver does not have any settings to setup".
9 choose your built unity application. If you built a screen saver with this app before, the editor script will show a message about replacing the existing screen saver, choose yes if you want to replace.
10 screen savers need to be placed in %windir% or %windir%/system32 to be recognized. You can copy them to your %windir% for testing but don't forget to create an installer for users. %windir% is the window's installation directory c:\windows on most systems but it's not guaranteed so use %windir%.

11 to create the installer you can use simple archivers like winrar or installation creator programs like nullsoft. I might create an installer creator for unity later.

Your screen saver is ready! You don't need to read the remaining parts of this document if you don't need to know the internal workings of this package.

What is a Screen Saver program?

a screen saver is a normal executable file that it's extension is changed to .scr. -woops why did I bought this- wait, wait, wait,
if you put a file with .scr extension in %windir% directory or %windir%/system32 directory which is c:/windows in most setups then it will be identified as a screen saver. You can choose it in screen saver applet of the control panel. Windows will send different command line arguments to the screen saver based on the intended behavior. When you press the preview button or when windows want to show the screen saver after computer is idle for some time it will send /s to your screen saver. when it want you to draw the screen saver in the small monitor window inside the screen saver applet of the control panel, it will send /p and the window handle of the window that you should draw the screen saver on it. When it want you to show the settings dialog of the screen saver (when user presses the settings button of the screen saver applet), it will send /c to your screen saver. A screen saver has another special behavior too. All screen savers will exit after user moves the mouse or presses any button. You might change it a little for fun but generally it should be easy to quit the screen saver.

How does it all work?

As you might guess, it's not that hard. We should get the command line arguments and process them. You are right but there are multiple problems.

1 unity games are really slow at startup because they are big executables and it will take some amount of time to run them so even if you want to exit the screen saver because the command line arg is not /s, unity game will open and user can see it. Then it will close but it's not something that you want for a real screen saver.

2 unity users (not pro) can not process command line arguments before unity's splash screen. So the unity game will open and show the unity logo for few

seconds then it will run your code and understand that it should exit.

3 we can not draw our screen saver on the small monitor inside the screen saver selector applet. Unity doesn't support drawing the game window on any window other than its own so this is not possible for now. If you really want it for this usage or any other, request it in feedback.unity3d.com it's not that hard to implement.

4 most of the times screen savers will run in the current screen resolution but we can not tell unity apps to don't change the resolution. They are ruder than what you might think because we can not even do it in command line arguments so forget about not changing the resolution. So add this to 1 and 2 because users will see 2 screen resolution changes when they just choose your screen saver in the drop down.

There are 2 ways for solving 1 and 2. The first way is to have a hook in unity before loading any graphics. If unity would have a function like OnApplicationStart () for us to program we could solve it easily. in that way unity would call our OnApplicationStart before loading any window and then we could check command line args and exit or continue based on them or even check the screen resolution of the system and run a full screen game in that resolution. Currently it's not possible but one might add it to feedback to. I might do it myself. The other way is to create a small starter program outside of unity. Then we will change the extension of that program to scr and then in that program we will check command line arguments and run the big, heavy and rude unity game if really needed. We use this approach in our package. We wrote the app in C# and .net 2 is required but writing the app in C++ is not that hard too. It's not needed because all vista/7 systems can run the systems and nearly all XPs have .NET 2 or higher installed. SP3 has .NET so they all have it. The main reason of this decision is that I currently do not have any C++ compiler installed on my system and you know that programmers are lazy. It's a basic checking of command line and using ShellExecute command. Windows has a header file for creating screen savers too but that's out of scope of this document and package. We can not solve 3 and 4 without help of unity technologies so forget them. They are not that important. Most screen savers use windows forms for their settings dialogs but if you want to do that then the starter app will become more complex and less generic. Unity GUI is complex enough and we can use PlayerPrefs to store data easily so we just tell the unity game to load the settings scene when tell it to run. We could use other approaches like having an appname_settings.exe and call it when we see /c but we choused this one.
The starter app is not a console app because they show a black console window for a second even if we close them soon. We used a windows forms application

but in the entry point we do all of our work and exit the application before running any form. If you never saw a C# windows forms application or if you don't know that where is the entry point. Open a windows forms app in visual studio or C# express and then open program.cs, it has a main function that is the real entry point of the application. We deleted those lines about running the form and initializing the application and put our code there. The source code of the simple starter is available in the downloaded package for you to take a look or modify. There is a built version inside the unity package so you don't need to touch the source code unless you want it yourself. The code basically does this. If command line is /s it will run the unity application, if it's /c then it will run the unity app and send /c command line argument to it. There is another version which is used for screen savers without settings. It will show a message box if it sees /c, the message box tells "this screen saver has not settings to setup.".
Now the unity game is running and it's not the end. We should check the command line arguments and if /c is sent we should load the settings scene, if not run the normal screen saver. We should have a code in our main scene to exit the screen saver when the user uses the mouse or keyboard. Our unity game should not display resolution dialog and this should be disabled in player settings before building.

What we have other than the starter app?

We have an editor script and a unity prefab and controller script.
The prefab is the controller script attached to an empty gameObject.
The controller script checks the command line and run settings scene if needed.
It also based on your selection will exit the screen saver with keyboard/mouse events.
The editor script just will ask you about your unity exe file and then copy a renamed version of starter near it. We could hide this all behind dlls but we put them as source code so you can learn from them. They are not complex scripts by no means but beginners and artists can learn from them. All unity users are not programmers and unity is all about making anyone enabled to do this 3D thing. We should help them too.

Scripts

The controller script is not complex and does not need any additional scripts. It has a small problem. If keyboard is enabled but mouse is disabled and user presses the mouse keys and keyboard keys together in the same frame, it will

not exit. But hey first of all it's not that easy to do (I tried it) and why users should do this. You will not do karate with your computer if you want to exit the screen saver, do you? As you see getting command line arguments sent to the process (your game) is easy, using GetCommandLineArguments you will get the array of command line arguments. The first element always is the name of your program. If the length of the array is larger than 1 it means that we have other arguments too. Arguments are separated by spaces and some other rules. Go to MSDN to read more about them.
The editor script just uses Directory and File classes of system.IO namespace to copy files and check their existence. It also runs the explorer app with the directory of our application's path to open its folder. Its easy, just creates a process class and tells it the filename and command line arguments to send and then starts the process. It's a nice wrapper around CreateProcess API.

Example scenes

There are two scenes called "main" and "settings" inside screen saver builder folder. These scenes are example empty scenes. The "main" scene just contains the screen Saver prefab with default values.

Example screen saver

The provided example is a simple screen saver which in it some color balls will be created and change their color over time and then destroy. It's just an example but I think it's beautiful.

Package content

Source: contains the source code of the starter program in it's both forms

Example contains the example screen saver

Documentation. PDF is this document

Questions and suggestions

If you have any questions regarding this document and package or any other thing about us or MindHammerGames contact us at info@mindhammergames.com please tell us about the quality of this document and package and readability of script files. This store is new so tell us what you think about the price. If you think there is a tool that can be built witch is useful for many developers/artists tell us to make the tool for you. Continuation of projects like this depends on sails of these packages and our revenue so don't copy or pirate our stuff please.

About MindHammerGames

MHG is a small game development team in Iran. We are the first web 3D developers in Iran and maybe in the entire Middle East. We use unity as our development tool and we are happy about that. We do any kind of development services from complete production to coding and optimizing other's code. Visit our website at http://en.mindhammergames.com/ for more information.

About Ashkan Saeedi Mazdeh

I live in Iran and born in Tehran. I love low level coding and my interest areas are game development, engine development, languages, image processing, computer graphics, parallel computing and distributed computing. I love other subjects like poetry, philosophy and ... I love studying scientist's ideas and religious beliefs. Like most other programmers, I try to look things differently and take a look at each possibility. There are many people that I love their ideas but Ben Franklin, Newton, Pascal, Einstein, Hafez, rumi and sa'di are my favorites. Currently I work as one of the members of MindHammerGames as a coder. In fact my friend Sina and I founded MindHammerGames last year.