The Ants Must Die: Ultra Ed. Documentation

Summary

The Ants Must Die: Ultra Edition (AMD:UE) is an ActionScript 2.0 (AS 2.0) game written for the Adobe Flash 8 platform. This document describes the structure of the game and the flow of the program. This document will not go into the details of the game code, since the code is already heavily documented.

List of Files

AntsMustDie.fla – Source File

AntsMustDie.swf – Compiled Shockwave Flash File

AntlionController.as – ActionScript 2.0 Source Code File for the class that controls the antlion

BlurCrossFadeEffect.as – ActionScript 2.0 Source Code File for the class that applies an effect

GameController.as – ActionScript 2.0 Source Code File for the class that controls the game state

Documentation.pdf–A distributable version of this documentation

content – Folder containing images used in the flash source file

- This folder contains many pictures embedded in the flash source file, as well as the sound file which is not. Since the sound file is not included in the flash file, **this folder is required to play the game with sound**. However, the images are merely provided for future modifiers' convenience.

Requirements:

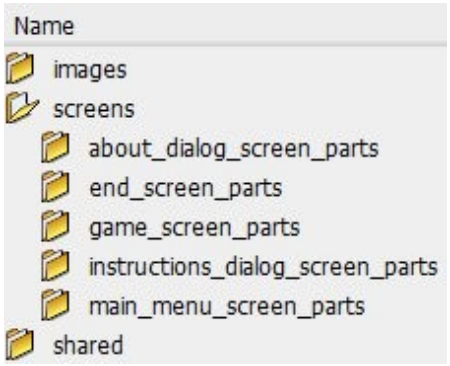
Playing this game requires an AdobeFlash 8 compatible player which supports ActionScript 2.0. The flash document was created in Flash 8, but can be edited in the newer Flash CS3 as well. To run the game, you can launch the "AntsMustDie.swf" file in the standalone flash player if it is installed on your system. Alternatively, you can open the same folder in any web browser that has the Flash 8 (or newer) plug-in installed, or you can compile and run the game from within the editor after opening the "AntsMustDie.fla" source file in AdobeFlash 8 or Flash CS3.

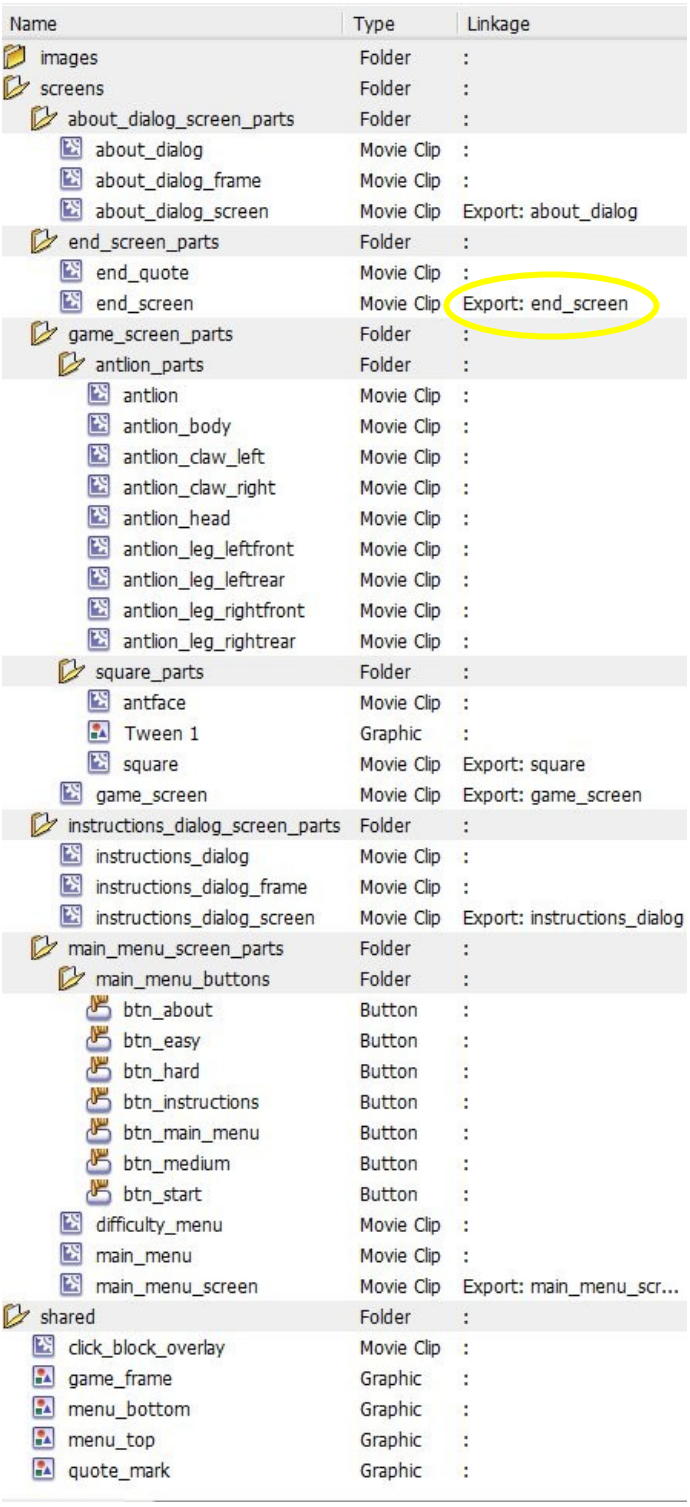
Potential Issues:

Do not add "Version 2 Components" such as the TextArea control (not to be confused with the lighter TextField control) to the flash document,as their mere presence in the document library will interfere with the proper functioning of the depth management functions used in ActionScript 2.0. See the notes at < http://livedocs.adobe.com/flash/8/main/00002510.html > for details on this behaviour

Document Library Contents

In the AMD:UE flash document's library(press F9 in flash), there are three top-level folders. Two contain some simple or animated graphics used here and there, so don't worry about them. The "screens" folder is important. It contains five subfolders, one for each kind of screen or sub-screen that is displayed to the player. These six folders contain the parts used to build each screen.





Library items which may be referenced in AS 2.0 must be exported with an identifier. In the image to the left, under the "Linkage" column, you can see which movie clips are exported for AS 2.0 along with their AS 2.0 identifier.

In AS 2.0, items from the library are added to parent movie clips at run-time by calling the *attachMovie*function, passing it four arguments:

1. The AS 2.0 identifier of a movie clip in the library
2. A new instance name for referencing the new instance in AS 2.0
3. A depth value
4. An initialization object for setting the new movie clip instance's initial property values.

Since the stage is initially empty, one can correctly reason that all content visible in the game must be added through AS 2.0, either directly by attachMovie, or indirectly by being a child of a clip that is added via attachMovie.

MovieClip Hierarchy

MovieClips in the flash player are structured hierarchically. The stage is initially empty in the Antlion source file, so the first objects to be added to the stage *at runtime* are always those exported for AS 2.0. MovieClips are added via the attachMovie function, and all instances have been given capitalized instance names, such as *MainMenuScreen*. MovieClips that are children of the exported MovieClips are also given capitalized instance names.

All MovieClips have at least one and up to three identifiers:

Library name
: All MovieClips have an entry in the library and therefore have a library name; this name is never used in scripting.

AS 2.0 identifier
: Any MovieClip added directly to the stage at runtime via the attachMovie function must have an AS 2.0 identifier to distinguish it from all other MovieClips exported for AS 2.0 from the library.

Instance name
: Every MovieClip on the stage must have an instance name to distinguish it from other MovieClips on the stage. This includes both MovieClips added in the editor as well as those added via the attachMovie function. The attachMovie function forces you to specify an instance name, but MovieClips that are added in the editor have optional instance names (and if you

The list on the following page shows the MovieClip hierarchy used in the game for MovieClips that have instance names and may be referenced in AS 2.0. Each line gives a MovieClip's instance name on the stage, then in parenthesis its library name. MovieClips exported for AS 2.0 also have their AS 2.0 identifiers (highlighted in green) in parenthesis after their instance name.

MovieClips exported for AS 2.0and added directly via the attachMovie function have been highlighted in yellow and green. All other (non-highlighted) items are MovieClips that are added indirectly as a result of being children of the exported MovieClips. Regardless of how a MovieClip is added (directly or indirectly), they can all be referenced in AS 2.0 with the appropriate instance name.

Referencing MovieClips in AS 2.0

To reference a MovieClip from a non-local scope, you must build absolute references to it, starting at the "_root" MovieClip and using dot notation to traverse the display hierarchy. For example, the "**DifficultyMenu**" can be referenced as "**_root.MainMenuScreen.DifficultyMenu**"; the "**antLion**" can be referenced as "**_root.GameScreen.antLion**". You can also reference non-MovieClips such as the "**GameController**" as "**_root.GameScreen.GameController**".

Keep in mind, however, that not all of these MovieClips are present on the stage at the same time, so their references will work only when they are on the stage. For example, "**_root.GameScreen**" and any child MovieClip is not valid until the player chooses a difficulty and the **GameScreen** is added via a call to attachMovie.

AMD:UE Instance Name Hierarchy (MovieClips and other classes)

_root

MainMenuScreen–(AS 2.0 ID: main_menu_screen; Library Name: main_menu_screen)

Instructions–(AS 2.0 ID: instructions_dialog; Library Name: instructions_dialog_screen)

- **Frame**–(Library Name: instructions_dialog_frame)

About–(AS 2.0 ID: about_dialog; Library Name: about_dialog_screen)

- **Frame**–(Library Name: instructions_dialog_frame)

Title–(Library Name: game_title)

MainMenu–(Library Name: main_menu)

- **Start**–(Library Name: btn_start)
- **Instructions**–(Library Name: btn_instructions)
- **About**–(Library Name: btn_about)

DifficultyMenu–(Library Name: difficulty_menu)

- **Easy**–(Library Name: btn_easy)
- **Medium**–(Library Name: btn_medium)
- **Hard**–(Library Name: btn_hard)
- **MainMenu**–(Library Name: btn_main_menu)

GameScreen–(AS 2.0 ID: game_screen; Library Name: game_screen)

GameController–(Instance of the GameController class)

- **“0 0” to “5 4”**–(AS 2.0 ID: grid; Library Name: square)

AntlionController–(Instance of the AntlionController class)

a**ntLion**–(Library Name: antlion)

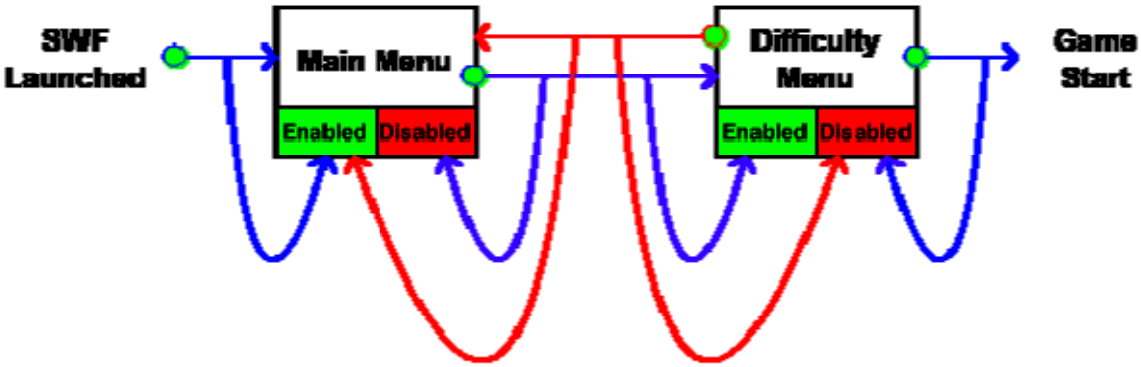
OutcomeScreen(AS 2.0 ID: lose_screen; Library Name: lose_screen)

Program Flow

The following is a description of the program flow. Many details that are left out of this description can be found among the code comments in the flash source file and AS 2.0 code files.

MainMenuScreen

When the application starts, a few settings are initialized, and the **MainMenuScreen** is added to the stage. The **MainMenuScreen** has the **MainMenu** and **DifficultyMenu** MovieClips as children. The player can explore the main menu options, and once the **Start** button is clicked, the **MainMenu** is disabled and the **DifficultyMenu** is displayed. At this point, the player can either choose a difficulty level, or return to the **MainMenu**.



GameScreen

Once a difficulty level is chosen, it is assigned to a global variable (**_global.GameDifficulty**), and the **GameScreen** is added to the display, starting the game. An instance of the **BlurCrossFadeEffect** class is created to transition from the **MainMenuScreen** to the newly added **GameScreen**. Once the transition is complete, the **MainMenuScreen** is removed, leaving only the **GameScreen**.

When the **GameScreen** was added, a few things happened. Instances of two other classes were initialized on the **GameScreen**'s first frame: **AntlionController** and **GameController**.

AntlionController

The **AntlionController**is autonomous. It is passed a reference to the **antLion** MovieClip (**_root.GameScreen.antLion**) and handles the **antLion**'s behavior for the remainder of the game.

GameController

The **GameController** maintains the state of the game. The **GameController** contains and array of **square** MovieClips (grid) which it places onto the **GameScreen** (_root.GameScreen) to make a grid. The **GameController** also keeps track of the time left over in the game, and the player's score when they click a square with an ant in it.

Square and the Movement Cycle

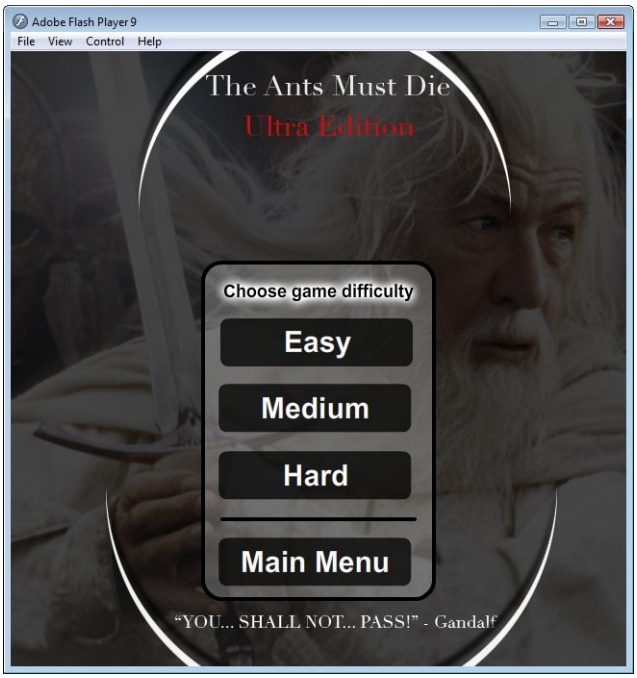
The **GameController** runs a simple cycle which is dependant on the time left on the clock. At different time Intervals, the **GameController** will choose one **square** at random (from grid) and instruct that square to show that it has an ant. The next time this cycle occurs, that square will be instructed to hide its ant while another square is chosen.

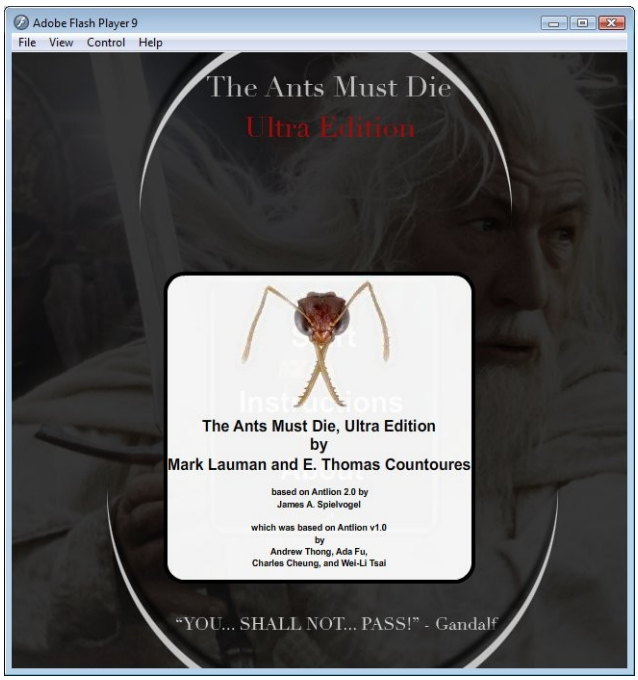
The **square** MovieClip contains a function that is called when the mouse is released over it (onMouseRelease). This function kills the ant if it is present, and then tells the **GameController** to add one point to the score.

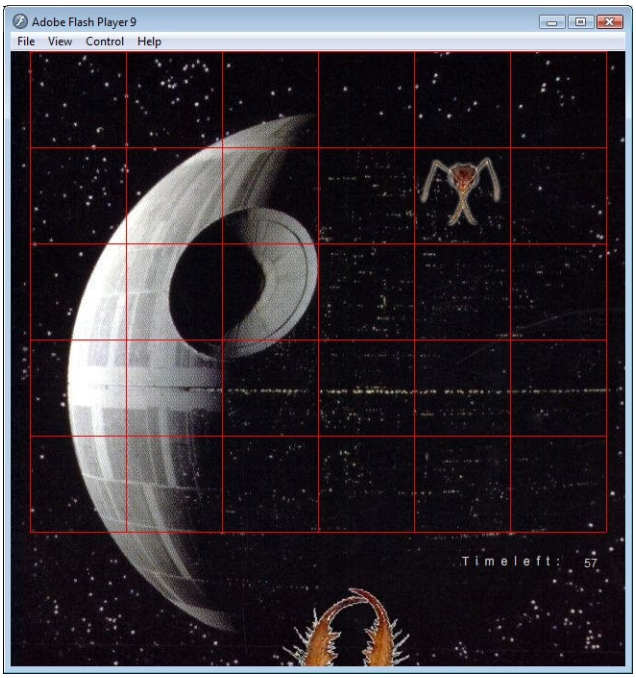
When the **GameController**'s timer has at last run out, the game is disposed and a new **BlurCrossFadeEffect** to the **OutcomeScreen** (an instance of end_screen)is initiated. The score is passed to the **OutcomeScreen** so that the player can know how well they did.

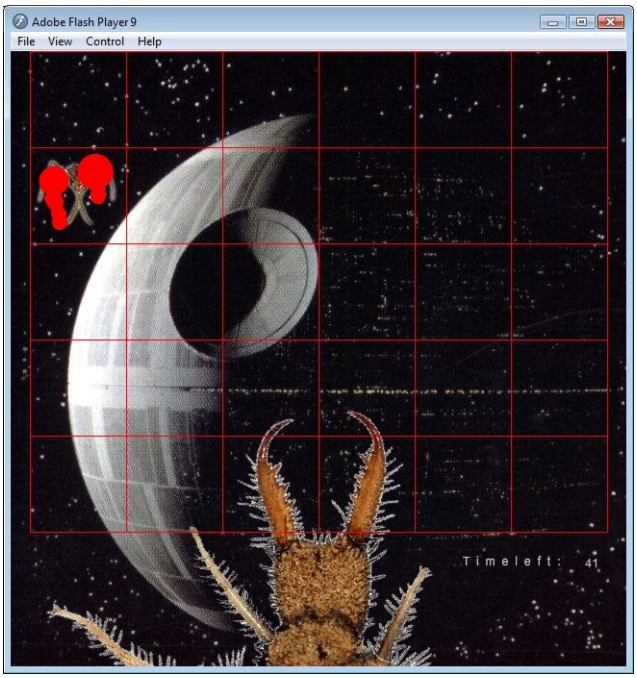
For more information, refer to the ActionScript code itself, which is heavily documented.

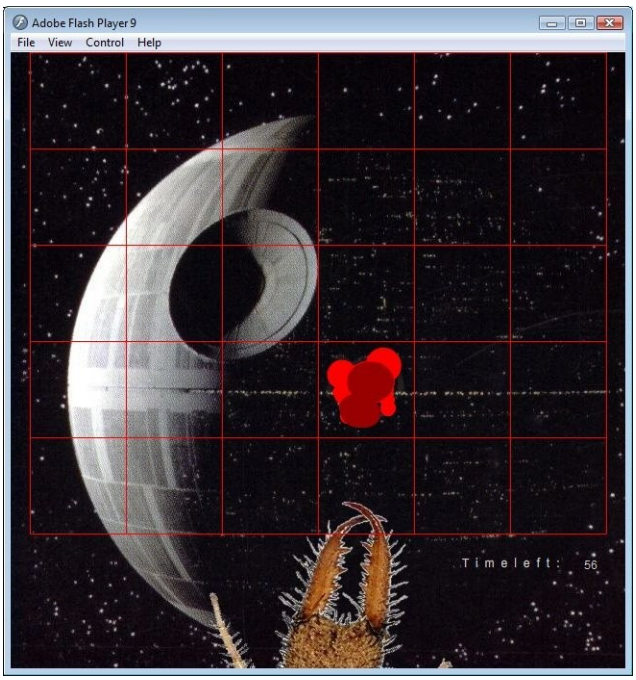
Screen Captures

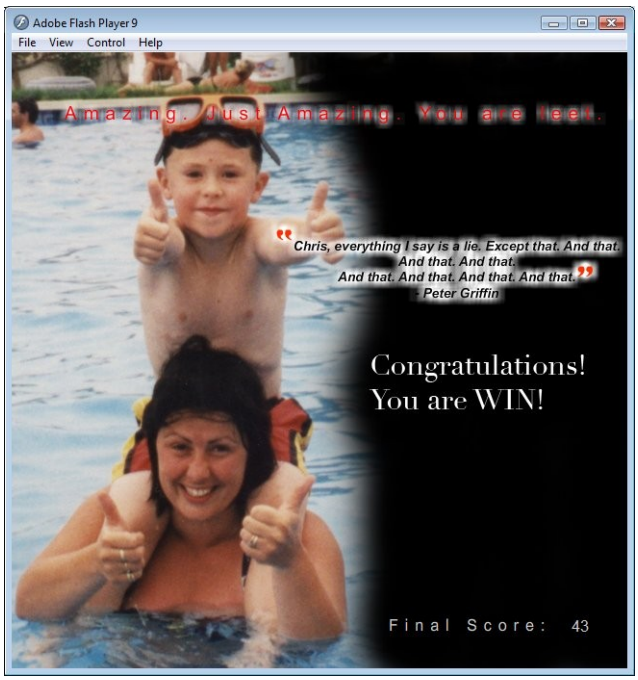





Adobe Flash Player 9
File View Control Help
Timeleft: 57


Adobe Flash Player 9
File View Control Help
Timeleft: 41


Adobe Flash Player 9
File View Control Help
Timeleft: 56


Adobe Flash Player 9
File View Control Help
Amazing. Just Amazing. You are leet.
"Chris, everything I say is a lie. Except that. And that.
And that. And that.
And that. And that. And that. And that."
- Peter Griffin
Congratulations!
You are WIN!
Final Score: 43