# Tutorial #12 – Leaderboards

## Overview

This tutorial will show a few examples of making leaderboards for the game [Doom](<https://retroachievements.org/game/11256>). Doom is an extremely popular first person shooter that was ported to many systems and still has the strong fan base to this day. The main reason why Doom was selected for this tutorial is that it was one of the first games to have speedrun competitions. Doom can record gameplay from user inputs making a small file that can be replayed on any computer with doom installed. This prompted players to record how quickly they could finish a level and upload the recording to compare with other players. Eventually, various categories of Doom speedruns were defined and the speedrun community grew and expanded to other games.<br>

![Doom Title Screen](Doom\_Title.png)<br>



## Leaderboards

Leaderboards are a ranked global list of players and game statistics. Leaderboards can be based on internal game memory or condition hit counts. There are many types of leaderboards already on RetroAchievements and some the most common are:

* Speedruns – Fastest time to complete an objective (level speedrun Example #12A and game speedrun Example #12C)
* Score Attack – Highest score achieved
* Action Count – How many times a player performs a certain action (kill count Example #12B)

## Game Events

Game events are a group of conditions used to determine when an important event occurs in the game. It works best if game events have a delta check in them so that they only occur for one frame. The best way to build conditions for a game event is to create a series of test achievements for each game event you are interested in using, activate all the test achievements, check the box “Keep Active” to test for achievements that trigger more than expected, then play through the game normally while noting when achievements trigger. Common game events include:

* Start Game
* Beat Game
* Start Level
* Beat Level
* On Title Screen
* Demo Mode
* Game Over
* Continue Used
* Lost Life
* Lost Health

### Start Event

The start event is a group of conditions used to prime a leaderboard. Once the leaderboard is primed the user will be notified and the value of the leaderboard will be displayed. Multiple start events can be defined by putting each event into separate alts.

### Cancel Event

The cancel event is a group of conditions used to stop a primed leaderboard. When a leaderboard is canceled the value is not submitted and the player is notified that they failed the leaderboard. Multiple cancel events can be defined by putting each event into separate alts.

### Submit Event

The cancel event is a group of conditions used to submit the value of a primed leaderboard. When a leaderboard is submitted the value is recorded to the site and the player is notified that they completed the leaderboard. Multiple submit events can be defined by putting each event into separate alts.

## Leaderboard Value

The leaderboard value is what gets submitted to the website when successfully completing a leaderboard. The value can be an internal memory, a combination of memory values (ie. score memory is BCD and needs to be converted to a decimal value), or a count of hits. If it can be measured then it can be used as a valued board value.

## Lower is Better?

The ```lower\_is\_better``` setting effects how the leaderboards is sorted. If true then the leaderboard will be sorted by putting the player with the lowest score in the #1 ranking. If false then the leaderboard will be sorted by putting the player with the highest score in the #1 ranking. If you accidently set ```lower\_is\_better``` to the wrong value then you can change it later and the leaderboard will resort the entries accordingly.

## Format

The format setting only effects how the value is displayed and not the base value submitted. If you accidently set the ```format``` to the wrong value then you can change it later and the leaderboard will displayed with the new ```format```. See the [RATools Wiki](<https://github.com/Jamiras/RATools/wiki/Leaderboard-Functions>) for a list of possible formats.

# Example #12A: Level Speedruns

Speedruns are a measure of how long it takes a player to accomplish a goal. Most speedruns are for accomplishing a goal as quickly as possible however speedruns can have additional conditions that stipulate the difficulty level, collecting all the items, or other handicaps. If the game is popular enough then it will have various types of speedruns listed on [Speedrun.com](<https://www.speedrun.com/>). If the game isn’t listed feel free to come up with your own challenges. For the speedruns in this example the player must be on the difficulty Hurt Me Plenty (difficulty 2 of 4) or higher for the leaderboard to activate. It is best practice to have some leaderboards that are active for the default settings before adding more complicated leaderboards.<br>

![Doom First level](Doom\_Level\_1.png)<br>



```

// Doom

// #ID = 11256

// $077E10: Map ID 1

// 00 = Main Menu

function MapId() => byte(0x077E10)

MapLookup = {

0x0: "Main Menu",

0x1: "Hangar",

0x2: "Plant",

0x3: "Toxin Refinery",

0x4: "Command Control",

0x5: "Phobos Lab",

0x6: "Central Processing",

0x7: "Computer Station",

0x8: "Phobos Anomaly",

0x9: "Deimos Anomaly",

0xA: "Containment Area",

0xB: "Refinery",

0xC: "Deimos Lab",

0xD: "Command Center",

0xE: "Halls of the Damned",

0xF: "Spawning Vats",

0x10: "Hell Gate",

0x11: "Hell Keep",

0x12: "Pandemonium",

0x13: "House of Pain",

0x14: "Unholy Cathedral",

0x15: "Mt. Erebus",

0x16: "Limbo",

0x17: "Tower of Babel",

0x18: "Hell Beneath",

0x19: "Perfect Hatred",

0x1A: "Sever the Wicked",

0x1B: "Unruly Evil",

0x1C: "Unto the Cruel",

0x1D: "Twilight Descends",

0x1E: "Threshold of Pain",

0x1F: "Entryway",

0x20: "Underhalls",

0x21: "The Gantlet",

0x22: "The Focus",

0x23: "The Waste Tunnels",

0x24: "The Crusher",

0x25: "Dead Simple",

0x26: "Tricks and Traps",

0x27: "The Pit",

0x28: "Refueling Base",

0x29: "O of Destruction!",

0x2A: "The Factory",

0x2B: "The Inmost Dens",

0x2C: "Suburbs",

0x2D: "Tenements",

0x2E: "The Courtyard",

0x2F: "The Citadel",

0x30: "Nirvana",

0x31: "The Catacombs",

0x32: "Barrels of Fun",

0x33: "Bloodfalls",

0x34: "The Abandoned Mines",

0x35: "Monster Condo",

0x36: "Redemption Denied",

0x37: "Fortress of Mistery",

0x38: "The Military Base",

0x39: "The Marshes",

0x3A: "The Mansion",

0x3B: "Club Doom",

}

// $077dc8: Results Screen

// 01 = Activated (Also at Main Menu)

function ResultScreen() => byte(0x077dc8)

// $07814C: Game Timer 32-bit

// (Reset to 0 when the game load a level)

function GameTimer() => dword(0x07814C)

// $078258: Current Difficulty

function Difficulty() => byte(0x078258)

// $08656A: Loading...

// 10= On screen

function Loading() => byte(0x08656A)

// $09874d: Intermission Screen Background

// 01 = Activated

function IntermissionScreen() => byte(0x09874d)

// $098740: Last HUD Text ID (16-bit)

// 10b0 = All powerful mode on

// 10e0= Lots of goodies!

// 52c= You got the BFG9000! Oh, yes.

// 5bc= You got the shotgun!

// 5d4= You got the supershotgun!

// 54c= You got the chaingun!

// 564= A chainsaw! Find some meat!

// 584= You got the rocket launcher!

// 5a4= You got the plasma gun!

function LastHudText() => word(0x098740)

// $098744: HUD Text Timer in screen

function HudTextTimerInScreen() => byte(0x098744)

// $0A88AC: Cheats

// Bit1= God Mode

// Bit2= Full Map

// Bit3= Show All Things on the Map

// Bit5= Level Warping

// Bit7= Invisible Walls

function Cheats() => byte(0x0A88AC)

/// Helper Functions ////////////////////////////////////////////////////////////

// Start when mapid = current map, no intermission or load screen,

// on Hurt me Plenty difficulty or higher, and the game timer has started

function StartMap(map)

{

return MapId() == map &&

Difficulty() >= 2 &&

ResultScreen() == 0 &&

IntermissionScreen() == 0 &&

prev(GameTimer()) == 0 &&

GameTimer() == 2

}

// End when mapid = current map and the results screen is shown

function EndMap(map)

{

return MapId() == map &&

prev(ResultScreen()) == 0 &&

ResultScreen() == 1

}

// Cancel when the loading screen appears

function LoadScreen() => IntermissionScreen() == 1 && Loading() == 0x10

// Cancel when the loading screen appears, or any cheat are used

function GoodiesCheat() => HudTextTimerInScreen() == 1 && LastHudText() == 0x0010e0

function CheatsBitflags() => Cheats() != 0

/// Leaderboards ////////////////////////////////////////////////////////////

// Create a leaderboard for each map

for index in range(0x1, 0x3B)

{

leaderboard

(

MapLookup[index] + " Speedrun",

"Fastest time to complete " + MapLookup[index] + " on Hurt Me Plenty or above",

StartMap(index),

LoadScreen() || GoodiesCheat() || CheatsBitflags(),

EndMap(index),

always\_true(),

format = "FRAMES",

lower\_is\_better = true

)

}

```

# One Logic to Rule Them All

Notice that the above code creates a speedrun leaderboard for each level. The logic for each leaderboard is very similar with the only difference being the ```MapID()```. The code takes advantage of this similarity and creates all the leaderboards by looping through each possible ```MapID()```. The ```MapLookup[]``` dictionary used to create the Rich Presence is also useful here to convert the ```MapID()``` to its name for the title and description of the leaderboard.

## Start Event

The function ```StartMap()``` is used to prime the leaderboard. The passed map id ensures that the leaderboard is only active on one map. The event also checks that the player is on difficulty Hurt Me Plenty or higher, we are not on the result or intermission screen, and when the internal game timer steps from 0 to 2.

## Cancel Event

The leaderboard will cancel if the loading screen is shown which means the player died or quit. It will also cancel if any of the cheat codes were activated. To ensure fairness leaderboards should be aware of in game cheats and either cancel or never get primed to begin with while a cheat is active. In this case the cheats were put in the cancel event in case the player activates a cheat during the level.

## Submit Event

The leaderboard will submit when the results screen is shown for the current map. The result screen is only shown when the player completes a map so it is a good event to use for the leaderboard submission.

## Value

The value ```always\_true()``` is used to count the number of frames that the leaderboard has been active for. The ```GameTimer()``` memory value would have been a good candidate for the value as well however, unfortunately the timer is reset on the result screen so would always submit zero.

## Lower is Better?

In the case of speedruns leaderboard setting ```lower\_is\_better = true``` will mean that the leaderboard entries are sorted lowest to the highest with the player who completed the level in the least amount of time ranking the #1 spot.

## Format

Since the value is counting the number of frames a level took to complete it is displayed to the screen and the site using ```format=”FRAMES”```.

# Example #12B: Kill Count

Killing demons is not necessary to beating a level however, it does make it a lot easier to proceed through the map. Each map in the Playstation set has three stacked achievements; one for beating the level, one for beating the level on Ultra-Violent (difficulty 4 of 4), and one for beating a map with 100% secrets and 100% kills. As such the player will need to kill every demon on a map if they are going for a mastery of the game. This leaderboard will track how many kills the player accumulates in one life for either Doom (maps 1-30) or Doom II (maps 31-54). <br>

![Doom Slayer killing a Demon](Doom\_Slayer.png)<br>



```

// Doom

// #ID = 11256

// $077E10: Map ID

function MapId() => byte(0x077E10)

// $077dc8: Results Screen

// 01 = Activated (Also at Main Menu)

function ResultScreen() => byte(0x077dc8)

// $07814C: Game Timer 32-bit

// (Reset to 0 when the game load a level)

function GameTimer() => dword(0x07814C)

// $078258: Current Difficulty

function Difficulty() => byte(0x078258)

// $08656A: Loading...

// 10= On screen

function Loading() => byte(0x08656A)

// $086763: [8-bit] Password Screen (0x64 = on password screen)

function PasswordScreen() => byte(0x086763)

// $09874d: Intermission Screen Background

// 01 = Activated

function IntermissionScreen() => byte(0x09874d)

// $098740: Last HUD Text ID (16-bit)

// 10b0 = All powerful mode on

// 10e0= Lots of goodies!

// 52c= You got the BFG9000! Oh, yes.

// 5bc= You got the shotgun!

// 5d4= You got the supershotgun!

// 54c= You got the chaingun!

// 564= A chainsaw! Find some meat!

// 584= You got the rocket launcher!

// 5a4= You got the plasma gun!

function LastHudText() => word(0x098740)

// $098744: HUD Text Timer in screen

function HudTextTimerInScreen() => byte(0x098744)

// $0A8810: Current Health (16bits)

function Health() => word(0x0A8810)

// $0A88AC: Cheats

// Bit1= God Mode

// Bit2= Full Map

// Bit3= Show All Things on the Map

// Bit5= Level Warping

// Bit7= Invisible Walls

function Cheats() => byte(0x0A88AC)

// $0a88b4: Total Kills (32-bit)

// Reset when a level is loaded

function TotalKills() => dword(0x0a88b4)

/// Helper Functions ////////////////////////////////////////////////////////////

// Start when mapid = current map, no intermission or load screen,

// on Hurt me Plenty difficulty or higher, and the game timer has started

function StartMap(map)

{

return MapId() == map &&

Difficulty() >= 2 &&

ResultScreen() == 0 &&

IntermissionScreen() == 0 &&

prev(GameTimer()) == 0 &&

GameTimer() == 2

}

// End when mapid = current map and the results screen is shown

function EndMap(map)

{

return MapId() == map &&

prev(ResultScreen()) == 0 &&

ResultScreen() == 1

}

// Cancel when the loading screen appears, or any cheat are used

function GoodiesCheat() => HudTextTimerInScreen() == 1 && LastHudText() == 0x0010e0

function CheatsBitflags() => Cheats() != 0

// Submit when main menu is open

function MainMenu() => MapId() == 0x0

// Cancel when entering a password

function PasswordEntry()

{

return PasswordScreen() == 0x64 &&

prev(Loading()) != 0x10 &&

Loading() == 0x10

}

// Submit when player dies

function Death() => Health() == 0x0

// Submit when main menu is open

function MainMenu() => MapId() == 0x0

// Level name constants

firstDoom1Level = 0x1

firstDoom2Level = 0x1f

lastDoom1Level = 0x1e

lastDoom2Level = 0x36

// Created a kill array to track if multiple monsters were killed at once

function KillCounter(killMax)

{

killArray = []

for index in range(0,killMax)

{

array\_push(killArray, prev(TotalKills()) + index < TotalKills())

}

// Use the in-game timer for the time value

return tally(2147483647, killArray)

}

/// Leaderboards ////////////////////////////////////////////////////////////

leaderboard

(

"Ultimate Doom Slayer",

"Most demons killed in one life starting from the first level of Ultimate Doom on Hurt Me Plenty or above",

StartMap(firstDoom1Level),

GoodiesCheat() || CheatsBitflags() || PasswordEntry(),

Death() || MainMenu() || EndMap(lastDoom1Level),

KillCounter(64),

format = "VALUE",

lower\_is\_better = false

)

leaderboard

(

"Doom II Slayer",

"Most demons killed in one life starting from the first level of Doom II on Hurt Me Plenty or above",

StartMap(firstDoom2Level),

GoodiesCheat() || CheatsBitflags() || PasswordEntry(),

Death() || MainMenu() || EndMap(lastDoom2Level),

KillCounter(64),

format = "VALUE",

lower\_is\_better = false

)

```

# Counting Game Actions

These achievements uses the changes in the memory ```TotalKills()``` to count how many kills the player has accumulated. We could use the ```TotalKills()``` directly however, it is reset every level. So instead we count the total number of hits by how much the value ```TotalKills()``` changes each frame with the function ```KillCounter()```. We can get an accurate count of kills by making an array of delta checks that will add a hit if the kills has gone up by 1 or more, 2 or more, …, to ```maxKills``` or more each frame. The value of ```maxKills``` can be set to any value but it should be high enough that it will account for the player killing multiple demons in the same frame with a large explosion.

## Start Event

The function ```StartMap()``` is used to prime the leaderboard. In this case we only want to start on map ID 0x1 for Doom or map ID 0x1f for Doom II. The event also checks that the player is on difficulty Hurt Me Plenty or higher, we are not on the result or intermission screen, and when the internal game timer steps from 0 to 2.

## Cancel Event

The leaderboard will cancel if the player enables any cheats or starts entering a password.

## Submit Event

The leaderboard will submit on death, exiting to the main menu, or by beating the last map ID 0x1e for Doom or map ID 0x36 for Doom II.

## Value

The value submitted is a running count of the total kills using the ```KillCounter()``` function.

## Lower is Better?

In this case ```lower\_is\_better = false``` is used since we want to order the leaderboard from most kills to least kills.

## Format

The format for these leaderboards is “VALUE” since we want to display the actual value of the number of monsters killed by the player.

# Example #12C: Game Speedruns

While improving speeds on individual maps will improve your overall time the ultimate speedrun challenge is a full game play through.<br>

![Doom II First level](Doom\_II\_Level\_1.png)<br>



# Homework #12

Create a leaderboard from the beginning map to the last map for both Doom and Doom II.

## Useful Memory

You need to use the map ID at byte 0x077E10 to complete the homework. Here’s the values for all the maps:

| ID | Name | Notes |

|:----:|-----------------------------------|-----------------------------------|

| 0x0 | Main Menu | Show Map the title screen |

| 0x1 | Hangar | First Map of Doom |

| 0x2 | Plant | |

| 0x3 | Toxin Refinery | |

| 0x4 | Command Control | |

| 0x5 | Phobos Lab | |

| 0x6 | Central Processing | |

| 0x7 | Computer Station | |

| 0x8 | Phobos Anomaly | |

| 0x9 | Deimos Anomaly | |

| 0xA | Containment Area | |

| 0xB | Refinery | |

| 0xC | Deimos Lab | |

| 0xD | Command Center | |

| 0xE | Halls of the Damned | |

| 0xF | Spawning Vats | |

| 0x10 | Hell Gate | |

| 0x11 | Hell Keep | |

| 0x12 | Pandemonium | |

| 0x13 | House of Pain | |

| 0x14 | Unholy Cathedral | |

| 0x15 | Mt. Erebus | |

| 0x16 | Limbo | |

| 0x17 | Tower of Babel | |

| 0x18 | Hell Beneath | |

| 0x19 | Perfect Hatred | |

| 0x1A | Sever the Wicked | |

| 0x1B | Unruly Evil | |

| 0x1C | Unto the Cruel | |

| 0x1D | Twilight Descends | |

| 0x1E | Threshold of Pain | Last Map of Doom |

| 0x1F | Entryway | First Map of Doom II |

| 0x20 | Underhalls | |

| 0x21 | The Gantlet | |

| 0x22 | The Focus | |

| 0x23 | The Waste Tunnels | |

| 0x24 | The Crusher | |

| 0x25 | Dead Simple | |

| 0x26 | Tricks and Traps | |

| 0x27 | The Pit | |

| 0x28 | Refueling Base | |

| 0x29 | O of Destruction! | |

| 0x2A | The Factory | |

| 0x2B | The Inmost Dens | |

| 0x2C | Suburbs | |

| 0x2D | Tenements | |

| 0x2E | The Courtyard | |

| 0x2F | The Citadel | |

| 0x30 | Nirvana | |

| 0x31 | The Catacombs | |

| 0x32 | Barrels of Fun | |

| 0x33 | Bloodfalls | |

| 0x34 | The Abandoned Mines | |

| 0x35 | Monster Condo | |

| 0x36 | Redemption Denied | Last Map of Doom II |

| 0x37 | Fortress of Mistery | Bonus Map |

| 0x38 | The Military Base | Bonus Map |

| 0x39 | The Marshes | Bonus Map |

| 0x3A | The Mansion | Bonus Map |

| 0x3B | Club Doom | Bonus Map |

# Solution #12C: Game Speedruns

This example will create a speedrun leaderboard from the beginning map to the last map for both Doom and Doom II.<br>

![Finishing a Map in Doom](Doom\_Exit.png)<br>



```

// Doom

// #ID = 11256

// $077E10: Map ID

function MapId() => byte(0x077E10)

// $077dc8: Results Screen

// 01 = Activated (Also at Main Menu)

function ResultScreen() => byte(0x077dc8)

// $07814C: Game Timer 32-bit

// (Reset to 0 when the game load a level)

function GameTimer() => dword(0x07814C)

// $078258: Current Difficulty

function Difficulty() => byte(0x078258)

// $08656A: Loading...

// 10= On screen

function Loading() => byte(0x08656A)

// $086763: [8-bit] Password Screen (0x64 = on password screen)

function PasswordScreen() => byte(0x086763)

// $09874d: Intermission Screen Background

// 01 = Activated

function IntermissionScreen() => byte(0x09874d)

// $098740: Last HUD Text ID (16-bit)

// 10b0 = All powerful mode on

// 10e0= Lots of goodies!

// 52c= You got the BFG9000! Oh, yes.

// 5bc= You got the shotgun!

// 5d4= You got the supershotgun!

// 54c= You got the chaingun!

// 564= A chainsaw! Find some meat!

// 584= You got the rocket launcher!

// 5a4= You got the plasma gun!

function LastHudText() => word(0x098740)

// $098744: HUD Text Timer in screen

function HudTextTimerInScreen() => byte(0x098744)

// $0A88AC: Cheats

// Bit1= God Mode

// Bit2= Full Map

// Bit3= Show All Things on the Map

// Bit5= Level Warping

// Bit7= Invisible Walls

function Cheats() => byte(0x0A88AC)

/// Helper Functions ////////////////////////////////////////////////////////////

// Start when mapid = current map, no intermission or load screen,

// on Hurt me Plenty difficulty or higher, and the game timer has started

function StartMap(map)

{

return MapId() == map &&

Difficulty() >= 2 &&

ResultScreen() == 0 &&

IntermissionScreen() == 0 &&

prev(GameTimer()) == 0 &&

GameTimer() == 2

}

// End when mapid = current map and the results screen is shown

function EndMap(map)

{

return MapId() == map &&

prev(ResultScreen()) == 0 &&

ResultScreen() == 1

}

// Cancel when the loading screen appears, or any cheat are used

function GoodiesCheat() => HudTextTimerInScreen() == 1 && LastHudText() == 0x0010e0

function CheatsBitflags() => Cheats() != 0

// Submit when main menu is open

function MainMenu() => MapId() == 0x0

// Cancel when entering a password

function PasswordEntry()

{

return PasswordScreen() == 0x64 &&

prev(Loading()) != 0x10 &&

Loading() == 0x10

}

// Level name constants

firstDoom1Level = 0x1

firstDoom2Level = 0x1f

lastDoom1Level = 0x1e

lastDoom2Level = 0x36

/// Leaderboards ////////////////////////////////////////////////////////////

leaderboard

(

"Ultimate Doom Speedrun",

"Quickest time to beat Ultimate Doom on Hurt Me Plenty or above",

StartMap(firstDoom1Level),

MainMenu() || GoodiesCheat() || CheatsBitflags() || PasswordEntry(),

EndMap(lastDoom1Level),

always\_true(),

format = "FRAMES",

lower\_is\_better = true

)

leaderboard

(

"Doom II Speedrun",

"Quickest time to beat Doom II on Hurt Me Plenty or above",

StartMap(firstDoom2Level),

MainMenu() || GoodiesCheat() || CheatsBitflags() || PasswordEntry(),

EndMap(lastDoom2Level),

always\_true(),

format = "FRAMES",

lower\_is\_better = true

)

```

## Start Event

The function ```StartMap()``` is used to prime the leaderboard. The Doom Speedrun leaderboard will start on map ID 0x1 and the Doom II leaderboard will start on map ID 0x1f. The event also checks the player is on difficulty Hurt Me Plenty or higher, we are not on the result or intermission screen, and when the internal game timer steps from 0 to 2.

## Cancel Event

The leaderboard will cancel if the player quits the game and returns to the main menu, enables any cheats, or starts entering a password.

## Submit Event

The leaderboard will submit when the results screen at the end of map ID 0x1e and the Doom II leaderboard the end of map ID 0x36. The result screen is only shown when the player completes a map so it is a good event to use for the leaderboard submission.

## Value

The value ```always\_true()``` is used to count the number of frames that the leaderboard has been active for. The ```GameTimer()``` memory value will not work for this leaderboard since it is reset every map.

## Lower is Better?

In the case of speedruns leaderboards setting ```lower\_is\_better = true``` will mean that the leaderboard entries are sorted lowest to the highest with the player who completed the game in the least amount of time ranking the #1 spot.

## Format

Since the value is counting the number of frames a game took to complete it is displayed to the screen and the site using ```format=”FRAMES”```.