Contents

[In-Text Citations 2](#_Toc169705667)

[Changes from Part 1 to Parts 2 and 3 4](#_Toc169705668)

[Methods 4](#_Toc169705669)

[Conditional/Ternary Operators 4](#_Toc169705670)

[Dictionaries 6](#_Toc169705671)

[Properties 6](#_Toc169705672)

[Lambda 6](#_Toc169705673)

[Other Changes 6](#_Toc169705674)

[GitHub Link 8](#_Toc169705675)

[Reference List 9](#_Toc169705676)

# In-Text Citations



Figure 1: Christian, B. 2022. Dragon Tactics: Follow These Tips for Truly Legendary Dragon Encounters.

(D&DBeyond, 2022)



Figure 2: Haitao, L. 2016. Dragon5.

(Art Station, 2016)



Figure 3: Ejdara. 2016. Wind Dragon

(Crunchyroll, 2016)



Figure 4: Tang, S. 2017. Forest Dragon.

(Art Station, 2017)

For in-text citations of code, see multi-line comment in the TurnPlayerForm class on **Lines 24-39**.

# Changes from Part 1 to Parts 2 and 3

## Methods

In the **StartGameForm** class there are two methods that were added to reduce the size of the class. Those are:

**SaveButtonClick(***…***)**

It takes in as parameters the player number, bool indicating that the player has saved as a ref, bool indicating that the opponent has saved, and all the GUI components related to player 1 that need to be customised. This includes the radio buttons to select the dragon type which will be used to check which dragon type is selected.

It is called on **Lines 68** and **74**. Which are within the methods called when player 1 and player 2 save respectively.

**UpdateColor(***…***)**

It takes in the background colour, foreground/text colour and a params array of GUI components as parameters. It then goes through all the GUI components with a for-each loop and sets their background and foreground colours to those specified in the parameters.

It is called on **Lines** **88-90**, **96**, **102**, **108**, **114**, **120**, **126**, **132** and **138**.

## Conditional/Ternary Operators

A conditional/ternary operator works as follows:

*[data type]* variable = *[condition to be tested]* ? *[value if true]* : *[value if false]*;

It can be written with an if-else statement as follows:

*[data type]* variable;

if (*[condition to be tested]* == true)

{

variable = *[value if true]*;

}

else

{

variable = *[value if false]*;

}

It can be found on **Line 84** in the **StartGameForm** class. It is a bit complicated so I will explain it with if-else statements.

It says:

string type = rbtnFireDragon.Checked ? FIRE\_DRAG\_NAME : rbtnIceDragon.Checked ? ICE\_DRAG\_NAME : rbtnWindDragon.Checked ? WIND\_DRAG\_NAME : rbtnEarthDragon.Checked ? EARTH\_DRAG\_NAME : "";

Written with if-else statements:

string type;

if (rbtnFireDragon.Checked)

{

type = FIRE\_DRAG\_NAME;

}

else

{

if (rbtnIceDragon.Checked)

{

type = ICE\_DRAG\_NAME;

}

else

{

if (rbtnWindDragon.Checked)

{

type = WIND\_DRAG\_NAME;

}

else

{

if (rbtnEarthDragon.Checked)

{

type = EARTH\_DRAG\_NAME;

}

else

{

type = “”;

}

}

}

}

It can be found on **Lines 52-53**, **55**, **65**, **76-77**, **79**, **92**, **103**, **130** and **158-161** and **199** in the **TurnPlayerForm** class. With **Lines 64** and **88** being unique, but following the same format, and the rest following the basic format.

**Lines 64** and **88** are written as follows:

blockMessage = (playerTurn == 1 ? p2IsBlocking : p1IsBlocking) ? " blocks it and" : "";

Written with if-else statements:

if (playerTurn == 1)

{

if (p2IsBlocking)

{

blockMessage = “ blocks it and”;

}

else

{

blockMessage = “”;

}

}

else

{

if (p1IsBlocking)

{

blockMessage = “ blocks it and”;

}

else

{

blockMessage = “”;

}

}

## Dictionaries

Dictionaries were added to the **StartGameForm** class to add a fast way to get data based off the dragon type.

**Lines 42-49** link the dragon type to its picture. It is used on **Lines 148** and **149** to save the picture of the selected dragon type to the **TurnPlayerForm** form.

**Lines 51-58** link the dragon type to an array of its values. It is used on **Line 85** to get the values of the selected dragon type to pass as arguments to the **SaveValues(***…***)** method on **Line** **86**.

## Properties

**Lines 13 and 15** in the **TurnPlayerForm** class are public properties that return the picture boxes for player 1 and player 2’s dragons respectively. They replace the **GetPicPlayer1Dragon()** and **GetPicPlayer2Dragon()** methods submitted in Part 1.

## Lambda

The lambda function on **Line 40** in the **TurnPlayerForm** class replaces the method that was used in Part 1 to force the application to close when the form is closed. This is done because the method contained only one line and never needed to be removed from the **FormClosing** delegate, so a lambda worked perfectly fine and reduced the size.

## Other Changes

On **Lines 156-168** in the StartGameForm class I set the arrays to be new arrays that contain the values in the parameters rather than saving them to individual indices in the arrays that was seen in the pseudocode for Part 1.

On **Lines 63-64**, **90-91**, **108-109**, **203-204** in the **TurnPlayerForm** class, rather than checking if it is the players turn and setting the bools accordingly, the bools are always set according to the player turn**OR** the bool’s current state.

The same is done on **Lines 87-88** in the **TurnPlayerForm** class, but it is for the player’s resting bool.

The same is done on **Lines 105-106** in the **TurnPlayerForm** class, but it is for the player’s blocking bool.

On **Lines 200-201** in the **TurnPlayerForm** class, the player’s resting bool is reset by doing an **AND** operation with the bool and if the player turn isn’t the opponents. This works because if the player is resting and it’s their turn (and they click the rest button) it will be **TRUE AND FALSE** which is **FALSE**, if they are not resting and it is their turn it will be **FALSE AND TRUE** which is **FALSE**, if they are resting and it is not their turn it will be **TRUE AND TRUE** which is **TRUE**, and if they are not resting and it is not their turn it will be **FALSE AND FALSE** which is **FALSE**. This means that in all cases it will reset the resting status to **FALSE** if it is their turn (and they click the rest button) or maintain their resting status if it is not their turn.

The same is done on **Lines 166-167** in the **TurnPlayerForm** class, but it is instead done for the blocking status which does the checks at the start of the turn (no buttons need to be pressed).

On **Line 153** in the **TurnPlayerForm** class, the line just sets both players “has played” status to false. “p1HasPlayed = p2HasPlayed = false;” is equivalent to just doing it over two lines:

“

p1HasPlayed = false;

p2HasPlayed = false;

“

On **Lines 162-165** in the **TurnPlayerForm** class rather than doing an if-else statement to show/hide buttons if the player needs to rest the code just sets the attack, special attack, and block button’s visibility to be the opposite of the player’s resting status (visible if they don’t need to rest and invisible if they do). The rest button works the same, but its visibility is the same as the player’s rest status (visible if they need to rest and invisible if they don’t)

# GitHub Link

<https://github.com/DavidBarker05/DavidTristanPOE>

# Reference List

Crunchyroll. 2016. *Wind Dragon*. [Online]. Available at: <https://img1.ak.crunchyroll.com/i/spire3/03302008/6/3/c/4/63c4ae8b94ca30_full.jpg> [Accessed 27 March 2024].

D&DBeyond. 2022. *Dragon Tactics: Follow These Tips for Truly Legendary Dragon Encounters*. [Online]. Available at: <https://www.dndbeyond.com/posts/1337-dragon-tactics-follow-these-tips-for-truly?epik=dj0yJnU9RXNzMktaUEdDSWxSQ2pwUEczUU90YVRoUFdOUGpDM3omcD0wJm49a1NfcGxiYmIwU3RYa0RCd19sb0QwUSZ0PUFBQUFBR1lGWjZ3> [Accessed 27 March 2024].

Art Station. 2016. *dragon5*. [Online]. Available at: <https://www.artstation.com/artwork/yKVBK> [Accessed 27 March 2024].

Microsoft. [s.a.]. Form.FormClosing Event (Version 8.0) [Source code]. <https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.form.formclosing?view=windowsdesktop-8.0#system-windows-forms-form-formclosing> (Accessed 27 March 2024).

Mircrosoft. [s.a.]. FormClosingEventHandler Delegate (Version 8.0) [Source code]. <https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.formclosingeventhandler?view=windowsdesktop-8.0#definition> (Accessed 27 March 2024).

Art Station. 2017. *Forest Dragon*. [Online]. Available at: <https://www.artstation.com/artwork/bPWzk> [Accessed 27 March 2024].

*What are Delegates? (C# Basics, Lambda, Action, Func)*. 2020. YouTube video, added by Code Monkey. [Online]. Available at: <https://www.youtube.com/watch?v=3ZfwqWl-YI0> [Accessed 10 May 2024].