Call of Duty Damage Calculator on Python and C++

Sebastián Constain Valencia

Duban Vargas Fula

Andres Ducuara

Pensamiento Algoritmico

Universidad Sergio Arboleda

06/03/2025

Contents

Introduction

The COD Damage Calculator is a project designed to calculate damage dealt when shooting through (wall-banging) different surfaces found in a COD environment, such as wood, metal and concrete. Its purpose is to help users compute damage based on different material types and apply relevant damage reduction calculations. Written in both Python and C++, this tool allows users to input base damage values and select a material type to determine the resulting damage after reduction. The project's dual-language approach makes it versatile and accessible to a wide range of users, including game developers, players, and researchers. The goal of the project is to provide a flexible and efficient tool for exploring damage scenarios in COD-style games.

Keywords: COD, Call Of Duty, Python, C++, Testing Environment

Problem Analysis

Description of the problem:

The goal of the project is to calculate the resulting damage of a bullet after passing through certain materials which are commonly found in a COD environment. The project will ask for a base damage and the material passing through, if the material is unknown but the damage dealt is, it will ask for the latter.

Functional and non-functional requirements:

Functional Requirements:

- User must be able to provide the base damage of the bullet
- User has to input the material type (Wood, metal, concrete or Unknown)
- The program should calculate the final damage of a bullet after being shot through a material
- If the material is unknown, the program should infer the material type based on the damage dealt and base damage.
- The program must display the result with 2 decimal points.

Non-Funcional Requirements:

- The solution should handle invalid inputs, ensuring the user provides valid numeric values
- The program should be efficient and produce the result quickly.
- Program must be easy to understand for a Begginer / intermediate programmer and easy to modify if new materials are added.

Main use cases:

1. Known Material:

User inputs the base damage and the material type (wood, metal or concrete). The program calculates the resulting damage after the damage reduction applied by the material.

2. Unknown Material:

User inputs the base damage and damage dealt after passing through a material. The program calculates the percentage reduction, compares it to the reduction rates of each material, and outputs the inferred material type.

Identification of Inputs, Processes, and Expected Outputs:

Inputs:

- Base damage of bullet (float)
- Material type (int: 1 for wood, 2 for metal, 3 for concrete, 0 for unknown)
- If the material is unknown, user will input the damage dealt after reduction (float)

Processes:

- If the material is known, apply the appropriate damage reduction formula for that material.

  e.g: if Material_type == 1, resulting_damage = base_damage – (0.3 * base_damage)

- If the material is unknown, calculate the percentage reduction and compare it with predefined values for each material type to infer the material.

  e.g: if material_type == 0, reduction_percent = (1- (dealt_damage / base_damage)) * 100

  if reduction_percent == 30: Material = “Wood”

Expected Outputs:

- If the material is known, display the resulting damage with 2 decimal points.
- If the material is unknown, display the inferred material type based on the reduction percentage.

Justification of solution

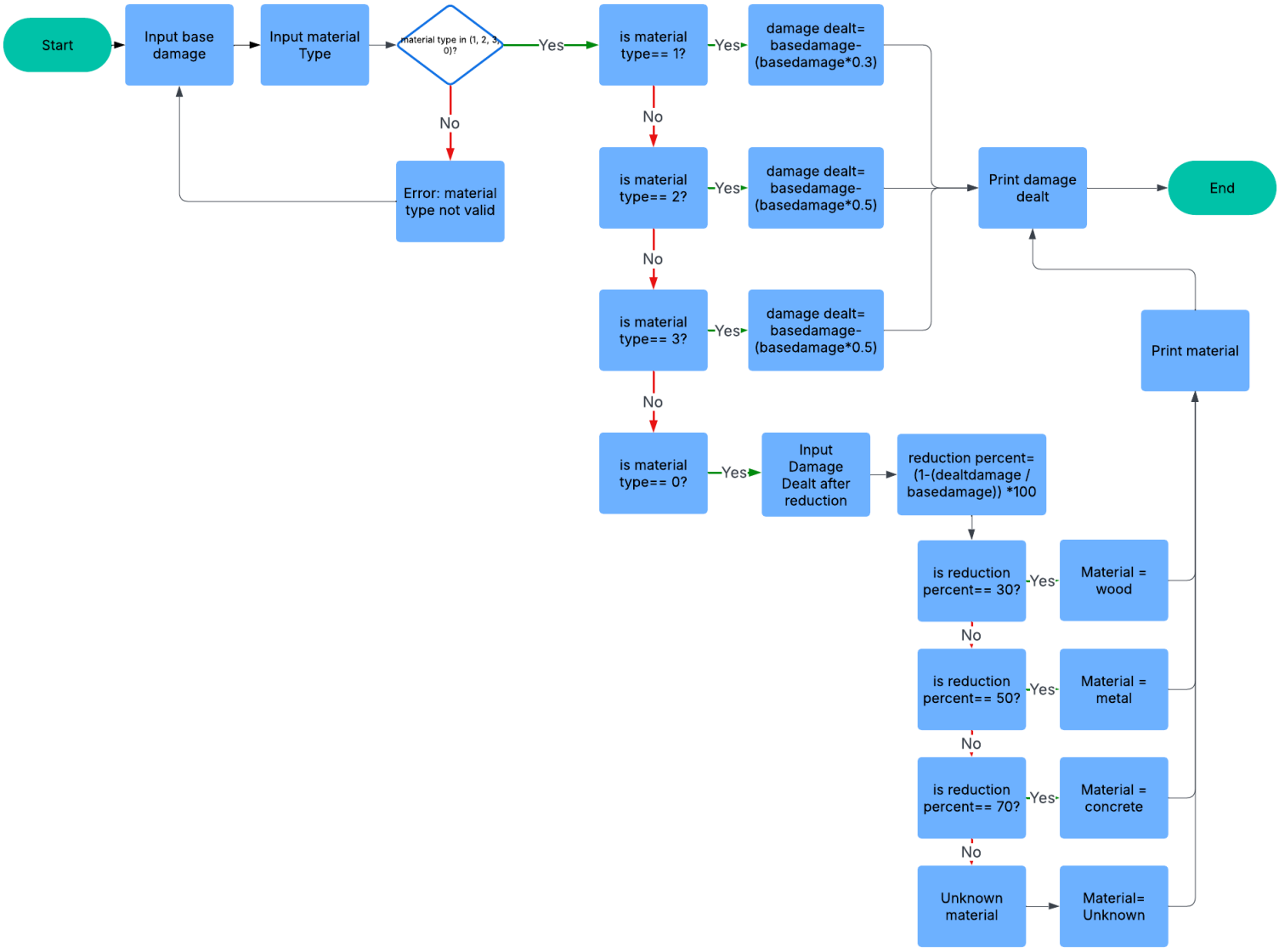
The program doesn't use any complex data structures like lists or dictionaries. It uses basic numeric inputs and conditional logic to handle material types. For this specific problem, using basic integers and floating-point numbers is sufficient because the operations are simple and direct. It also uses straight-forward operations to calculate the damage dealt or to infer the damage type.

This version of the code has a very straight-forward approach, using conditionals (if, elif and else) to handle different cases, and has basic error handling with the use of (try, except) for invalid input types. It's a functional and very simple approach that works in short-scale projects such as this one.

Comparison with Alternative Solutions and Final Choice:

An alternative would be to use Classes (Stroustrup, 2008), polymorphism (Blasco, 2019) and Lambda functions (Hutton, 2007), which, despite them being better alternatives for a bigger project and scalable project, it is not necessarily easier for small-scaled projects like this one, and using lambda functions in the code as is, wouldn't help organize or optimize the code much because of its simplicity. Apart from this, this code was made by first semester coders, and implementing classes and polymorphism would make the project harder to understand and modify if needed.

Flowchart


Start
Input base damage
Input material Type
material type in (1, 2, 3, 0)?
Yes
No
Error: material type not valid
is material type== 1?
Yes
damage dealt= basedamage-(basedamage*0.3)
No
is material type== 2?
Yes
damage dealt= basedamage-(basedamage*0.5)
No
is material type== 3?
Yes
damage dealt= basedamage-(basedamage*0.5)
No
is material type== 0?
Yes
Input Damage Dealt after reduction
reduction percent= (1-(dealtdamage / basedamage)) *100
is reduction percent== 30?
Yes
Material = wood
No
is reduction percent== 50?
Yes
Material = metal
No
is reduction percent== 70?
Yes
Material = concrete
No
Unknown material
Material= Unknown
Print material
Print damage dealt
End

Bibliography

Blasco, F. (2019). *Programación Orientada a Objetos en JAVA*. RA-MA Editorial.

Hutton, G. (2007). *Programming in Haskell*. Cambridge University Press.

Stroustrup, B. (2008). *Programming: Principles and Practice Using C++*. http://ci.nii.ac.jp/ncid/BB17977770