20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Dyanesh S
CB.EN.U4CYS21015
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

Outline

Paradigm 1 - Procedural

The procedural programming paradigm is characterized by a linear and sequential flow of control. It involves breaking down tasks into smaller procedures or subroutines, emphasizing a top-down, step-by-step problem-solving approach.

Key principles of the procedural paradigm:

- Modularity
- Sequential Execution
- Abstraction
- Pre-defined Functions
- Parameter Passing
- Local & Global Variables

Pascal

- Pascal is a high-level, procedural programming language designed by Niklaus Wirth in the late 1960s. It emphasizes readability, simplicity, and structured programming principles.
- It influenced the development of other languages and was popularized by implementations such as Turbo Pascal and Delphi. While not as widely used today, Pascal's legacy lives on through its impact on programming language design.
- Hello World Program in Pascal

```
program HelloWorld(output);
begin
    WriteLn('Hello, World!')
end
```

Pascal

Features of Pascal Language associated with procedural paradigm

- Pascal is a very structured language and uses the control structures like if-else, repeat-until statements, etc.
- It is having different data structures that are included with the records, arrays, files, pointers, etc.
- Pascal provides simplicity and provides a modular approach for machine implementation. It allows the features to be related to the compiler.
- Pascal uses minimum ambiguity to represent the data and its structure it is processed with some exceptions and provides smaller elements with their definitions.
- Pascal provides the exact sizes used by the operands and operators to perform on them. It provides a way to process and use the efficient code.

Paradigm 2 - Dataflow

Dataflow programming is a programming paradigm that represents the program as a directed graph, where nodes represent operations (functions or calculations) and edges represent data flowing between them. This contrasts with traditional imperative programming, which emphasizes sequential execution of commands.

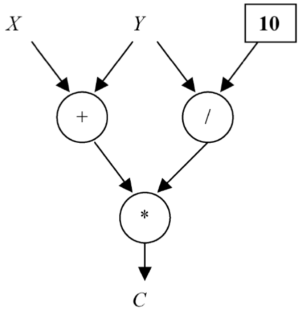
Components of Dataflow Paradigm:

- *Nodes (or Actors):* These represent processing units or operations. Each node performs a specific function or computation on the data.
- *Edges (or Channels):* These represent the flow of data between nodes. Data flows from one node to another through these edges.
- *Data Dependency:* Nodes only execute when the necessary input data is available. This leads to a more asynchronous and parallel execution model compared to traditional sequential programming.

Paradigm 2 - Dataflow

```
A := X + Y
B := Y / 10
C := A * B
```

(a)



(b)

Figure: A simple program and its dataflow equivalent

Paradigm 2 - Dataflow

Key concepts of Dataflow paradigm: [t]

- Concurrency and Parallelism
- Asynchronous Executio
- Modularity
- Explicit Data Dependencies
- Dynamic and Static Graphs
- No Global State
- Reactiveness
- Ease of Visual Representatio

Blender Game Engine

The Blender Game Engine (BGE) was a real-time interactive 3D game engine integrated into the Blender 3D modeling and animation software. It allowed users to create games directly within the Blender environment, providing tools for modeling, texturing, animating, and programming game logic.

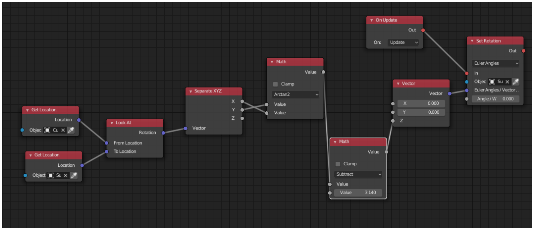


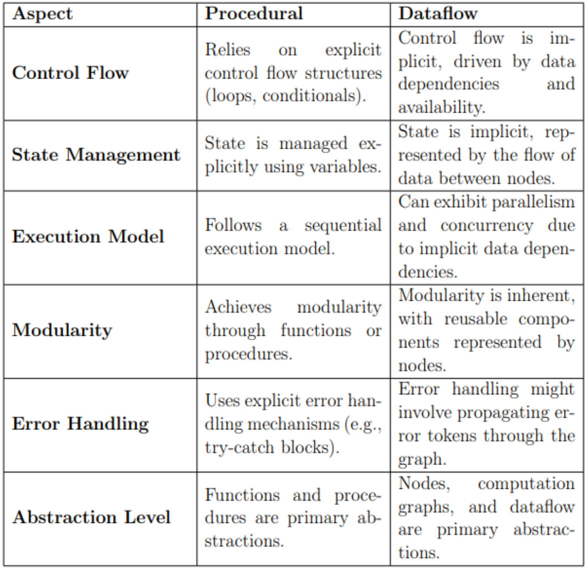
Figure: BGE Logic Node Editor

Blender Game Engine

- BGE provides the developers with logic brick editor to visually code com- plex game logic with the help of graphs and nodes. The Logic Brick Editor has three components:
  - Sensor Column
  - Controller Column
  - Actuator Column
- Along with Logic Bricks system, BGE offers Logic Node Editor which lets the developer connect one more nodes with others and construct complex game logics.
- There are many different types of nodes that can be utilized by developers to program the nodes to their needs, and some of the common ones are input nodes, output nodes, controller nodes, and scripting nodes.
- Logic Nodes seamlessly integrate with Python scripting using Python API that the BGE offers, allowing developers for more advanced and custom game behaviors

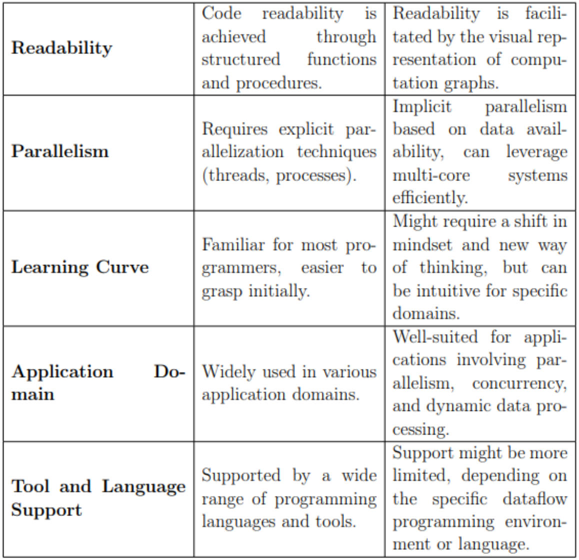
Comparisons

Similarities and Differences between Procedural and Dataflow



Aspect	Procedural	Dataflow
Control Flow	Relies on explicit control flow structures (loops, conditionals).	Control flow is implicit, driven by data dependencies and availability.
State Management	State is managed explicitly using variables.	State is implicit, represented by the flow of data between nodes.
Execution Model	Follows a sequential execution model.	Can exhibit parallelism and concurrency due to implicit data dependencies.
Modularity	Achieves modularity through functions or procedures.	Modularity is inherent, with reusable components represented by nodes.
Error Handling	Uses explicit error handling mechanisms (e.g., try-catch blocks).	Error handling might involve propagating error tokens through the graph.
Abstraction Level	Functions and procedures are primary abstractions.	Nodes, computation graphs, and dataflow are primary abstractions.

Comparisons



Readability	Code readability is achieved through structured functions and procedures.	Readability is facilitated by the visual representation of computation graphs.
Parallelism	Requires explicit parallelization techniques (threads, processes).	Implicit parallelism based on data availability, can leverage multi-core systems efficiently.
Learning Curve	Familiar for most programmers, easier to grasp initially.	Might require a shift in mindset and new way of thinking, but can be intuitive for specific domains.
Application Domain	Widely used in various application domains.	Well-suited for applications involving parallelism, concurrency, and dynamic data processing.
Tool and Language Support	Supported by a wide range of programming languages and tools.	Support might be more limited, depending on the specific dataflow programming environment or language.

References

1. https://hackr.io/blog/procedural-programming
2. https://learnloner.com/introduction-to-procedural/
3. https://www.careerride.com/view/what-are-the-features-that-make-pascal-a-good-language-in-modern-programming-2598.aspx/
4. https://devopedia.org/dataflow-programming
5. https://upbge.org/docs/latest/api/index.html
6. https://www.youtube.com/watch?v=yKKy0vJ0CrY
7. https://www.youtube.com/watch?v=u-uQqhpXIQA
8. Wikipedia
9. OpenAI Chat