



# CodeNect: Visual Programming Software for Learning Fundamentals of Programming

Brandon B. Lim-it, Jaykel O. Punay

May, 2021



# Introduction

- Technology is constantly progressing and improving



# Introduction

- Technology is constantly progressing and improving
- Programming is essential in the field of technology



# Introduction

- Technology is constantly progressing and improving
- Programming is essential in the field of technology
- Programming is a discipline



# Introduction

- Technology is constantly progressing and improving
- Programming is essential in the field of technology
- Programming is a discipline
- Programming is difficult



# Introduction

- Technology is constantly progressing and improving
- Programming is essential in the field of technology
- Programming is a discipline
- Programming is difficult
- Learning programming is more difficult



## Statement of the Problem

The fundamental concepts of programming are essential basics that are necessary for programmers to master. Concepts such as:

- Syntax and Semantics



## Statement of the Problem

The fundamental concepts of programming are essential basics that are necessary for programmers to master. Concepts such as:

- Syntax and Semantics
- Data Types and Data Structures





# Statement of the Problem

The fundamental concepts of programming are essential basics that are necessary for programmers to master. Concepts such as:

- Syntax and Semantics
- Data Types and Data Structures
- Logic and Conditionals



# Statement of the Problem

The fundamental concepts of programming are essential basics that are necessary for programmers to master. Concepts such as:

- Syntax and Semantics
- Data Types and Data Structures
- Logic and Conditionals
- Loops and Algorithm



# Statement of the Problem

The fundamental concepts of programming are essential basics that are necessary for programmers to master. Concepts such as:

- Syntax and Semantics
- Data Types and Data Structures
- Logic and Conditionals
- Loops and Algorithm
- Memory



## Statement of the Problem

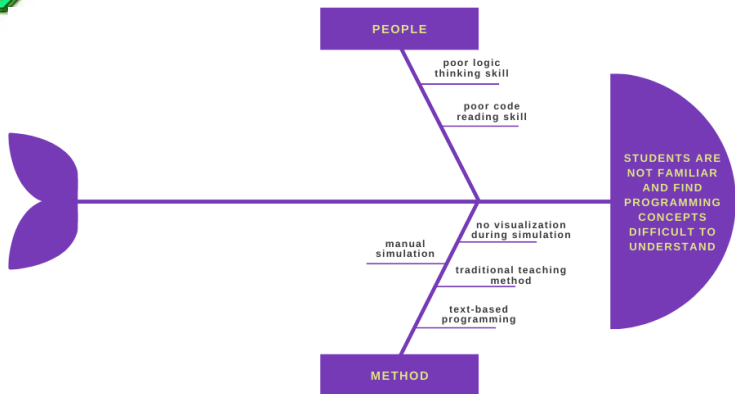
The fundamental concepts of programming are essential basics that are necessary for programmers to master. Concepts such as:

- Syntax and Semantics
- Data Types and Data Structures
- Logic and Conditionals
- Loops and Algorithm
- Memory

are key to easily understanding and getting better at programming as programming is a discipline (Prahofer, Hurnaus, Wirth, and Mossenbock, 2007).



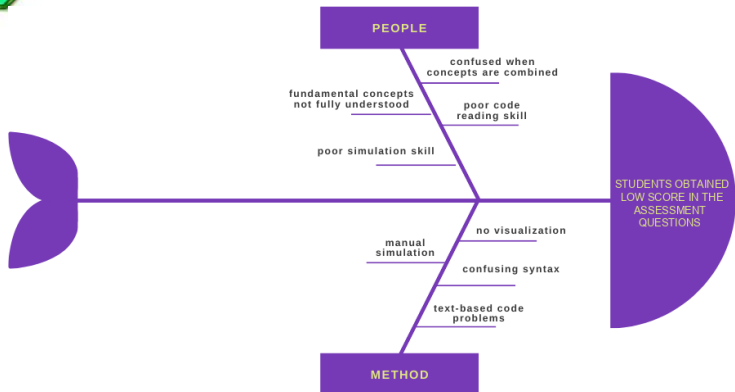
# Ishikawa Diagrams



Ishikawa diagram of students not familiar and finding programming concepts difficult to understand



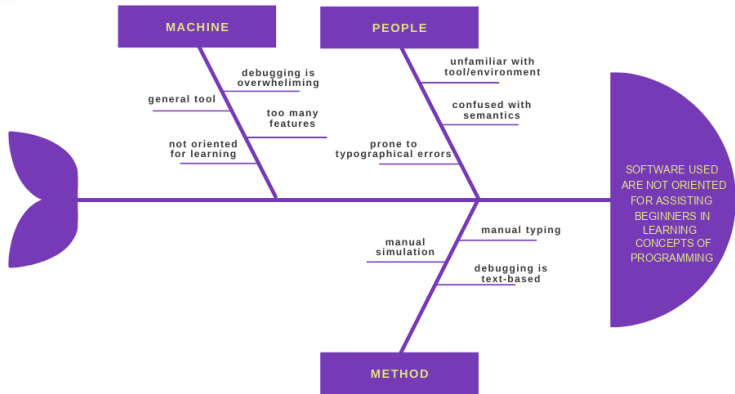
# Ishikawa Diagrams



Ishikawa diagram of the students incorrect answer to programming assessment



# Ishikawa Diagrams



Ishikawa diagram of the tools for programming not effective for learning



## Objectives of the Study

The general objective of the study is to develop a CodeNect: Visual Programming Software that will help in learning the fundamentals of programming.

Specifically, this study seeks:

- Identify the concepts learners find difficult to understand through conducted survey.





## Objectives of the Study

The general objective of the study is to develop a CodeNect: Visual Programming Software that will help in learning the fundamentals of programming.

Specifically, this study seeks:

- Identify the concepts learners find difficult to understand through conducted survey.
- Analyze the problems through a Ishikawa/Fishbone Diagram.



## Objectives of the Study

The general objective of the study is to develop a CodeNect: Visual Programming Software that will help in learning the fundamentals of programming.

Specifically, this study seeks:

- Identify the concepts learners find difficult to understand through conducted survey.
- Analyze the problems through a Ishikawa/Fishbone Diagram.
- Design the system using the Use Case Diagrams.



## Objectives of the Study

The general objective of the study is to develop a CodeNect: Visual Programming Software that will help in learning the fundamentals of programming.

Specifically, this study seeks:

- Identify the concepts learners find difficult to understand through conducted survey.
- Analyze the problems through a Ishikawa/Fishbone Diagram.
- Design the system using the Use Case Diagrams.
- Test the usability, functionality of the software using Experience-based test design.



## Objectives of the Study

The general objective of the study is to develop a CodeNect: Visual Programming Software that will help in learning the fundamentals of programming.

Specifically, this study seeks:

- Identify the concepts learners find difficult to understand through conducted survey.
- Analyze the problems through a Ishikawa/Fishbone Diagram.
- Design the system using the Use Case Diagrams.
- Test the usability, functionality of the software using Experience-based test design.
- Evaluate the acceptability of the software using the ISO 9126.



## Objectives of the Study

- Develop the software with the following main features:



## Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module



## Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module
  - ▶ Filesystem Module



# Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module
  - ▶ Filesystem Module
  - ▶ Input/Output Module





# Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module
  - ▶ Filesystem Module
  - ▶ Input/Output Module
  - ▶ Debug Module



# Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module
  - ▶ Filesystem Module
  - ▶ Input/Output Module
  - ▶ Debug Module



# Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module
  - ▶ Filesystem Module
  - ▶ Input/Output Module
  - ▶ Debug Module
  - ▶ Transpiler Module



# Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module
  - ▶ Filesystem Module
  - ▶ Input/Output Module
  - ▶ Debug Module
  - ▶ Transpiler Module
  - ▶ Assessment Module



# Objectives of the Study

- Develop the software with the following main features:
  - ▶ Visual Nodes Module
  - ▶ Filesystem Module
  - ▶ Input/Output Module
  - ▶ Debug Module
  - ▶ Simulation Module
  - ▶ Transpiler Module
  - ▶ Assessment Module



## Significance of the Study

### Students

The software helps in the education and improvement in the knowledge, skills, understanding, and expertise of the students and learners about programming. Thus, allowing them to compete and increasing the opportunities for their careers.

### Teachers

The software provides assistance for teachers and instructors to teach and demo programming concepts through visualization. This aids in relieving workload, stress, and maximizing lessons each class time.



## Significance of the Study

### Educational Institutions

The software benefits educational institutions like university for computer laboratory classes by providing a free software oriented for the purpose of learning

### Developers

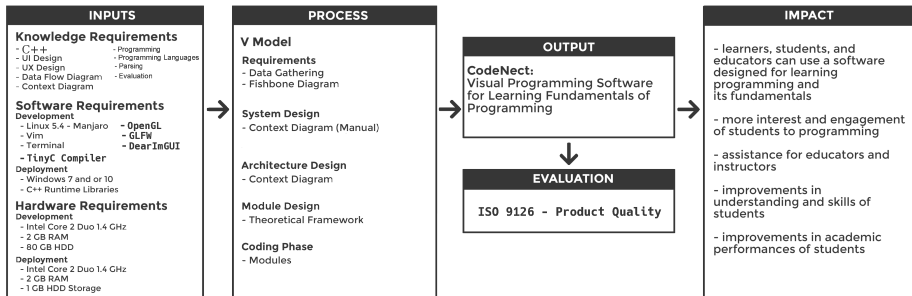
The software provides learning experience for the developers and researchers in preparation for software development career.

### Researchers

This study serves as a guide and reference in the field of software development and education for future researchers.



# Conceptual Framework of the Study







# Scope and Limitations of the Study

## Simplicity and Functionality

The software will prioritize simple and basic functionalities over numerous features for the purpose of learning and education.

## Stand-alone Program

The software will have no account management and can be run without any hassle. The software works perfectly in offline mode.

## Terminal-based

The software is limited to simulating text-based or command/terminal prompts as the priority is learning the fundamentals of programming.



# Scope and Limitations of the Study

## Visual Nodes Module

Nodes are graphical elements that serve as the building blocks of the software. Nodes can be used as a variable, logic, and conditionals

## Filesystem Module

Serves as the interface between the software and the user's machine for handling files such as creation, modification, reading, and deletion.



# Scope and Limitations of the Study

## Input/Output Module

The module is responsible for processing and responding events and performing actions based on the event such as key press, mouse click, and mouse movement.

## Debug Module

This module will linter and give feedback and indication to the user whenever there is an attempt to perform an action that is faulty in logic



# Scope and Limitations of the Study

## Simulation Module

The process of simulation involves the compiling, building, and running the visual code is executed by this module

## Transpiler Module

This module transpiles the visual code made by the user into source code in target programming language



# Scope and Limitations of the Study

## Assessment Module

The functionality of providing exercises designed for the learning of topics and concepts in programming and evaluation of the results are handled by this module



# Screenshots

## Visul Nodes Modules

- Basic Node ▶
    - Variable
  - Action Node ▶
    - Operation
  - Math Functions ▶
    - Cast
  - String Methods ▶
    - Comparison
  - Data Structure Node ▶
    - Branch
  - Get ▶
  - Loop ▶
  - Branch
- 
- Reset Zoom
  - Reset View Position

# Screenshots

## Visul Nodes Modules

**+ Create Node**

Variable Name

☒ **EMPTY**  Data Type

☐ **BOOL** Description (?)

☒ **INTEGER**

☐ **FLOAT**

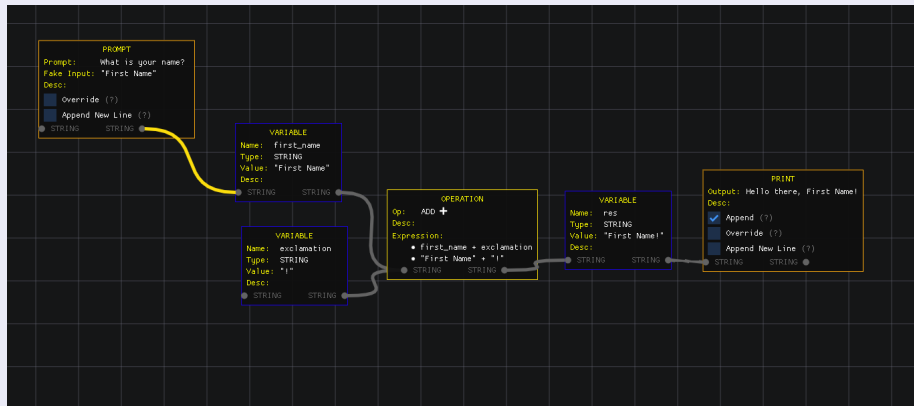
☐ **DOUBLE**

☐ **STRING**



# Screenshots

## Visul Nodes Modules

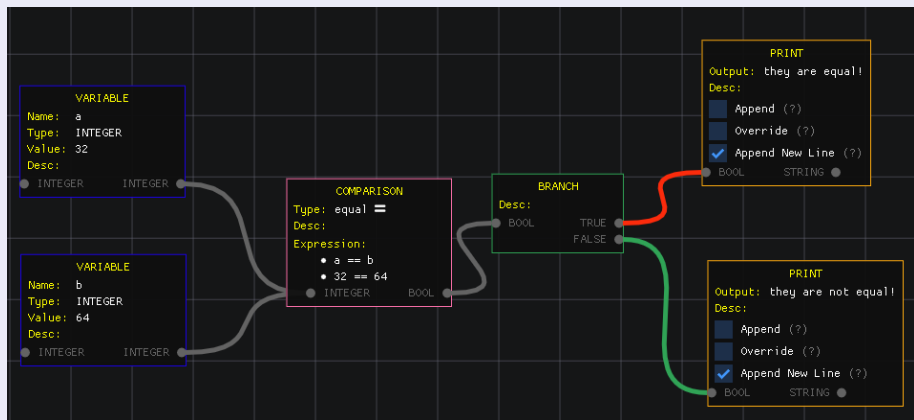






# Screenshots

## Visul Nodes Modules



# Screenshots

## Filesystem Module

Save As

« CodeNect... » bin\_win

Search bin\_win

Organize New folder

Name	Date modified	Type
assets	5/21/2021 9:40 PM	File folder
lib	5/21/2021 9:40 PM	File folder
tcc	5/21/2021 9:45 PM	File folder

File name:

Save as type: \*.cn

Save Cancel

NEW PROJECT

Filename Title Author

Create New Project Cancel

Hover on the left side to access the sidebar

Press <Ctrl+Shift+p> to access the command palette

Press <Ctrl+Shift+t> to open/hide the terminal

Press <Ctrl+Shift+i> to open/hide the inspector



# Screenshots

## Filesystem Module

File: /home/brbl/Projects/CodeNect/test/test\_assessment.cn

```
[meta]
title = Name
author = Brandon
creation_dt = 21-05-2021 11:58:07
offsetx = -47.000000
offsety = -126.000000
```

```
[node_0]
name = first_name
kind = VARIABLE
x = 71.600037
y = 295.000000
desc =
value_slot = STRING
value = What is your name? Brandon
input_1 = STRING
output_1 = STRING
```

```
[connection_PRINT_0_STRING-first_name_STRING]
in_node_name = PRINT_0
in_slot = STRING
out_node_name = first_name
out_slot = STRING
```

```
[node_1]
name = PRINT_0
kind = ACTION
x = 410.500000
y = 309.500000
desc =
action = PRINT
value =
```



# Screenshots

## Input/Output Module

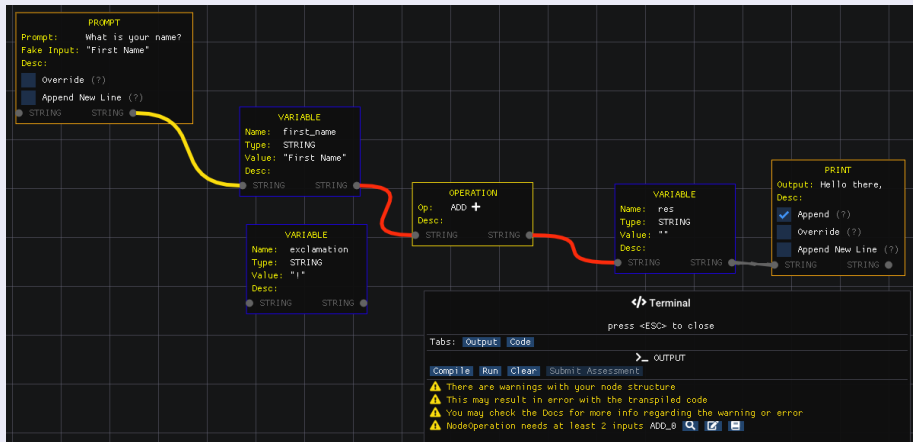
The screenshot illustrates the Input/Output module's functionality through three main components:

- Visual Programming:** A node-based editor with a 'PROMPT' node (Prompt: 'What is your name?', False Input: 'First Name', Desc: 'Override (?)', Append New Line: (?)) connected to two 'VARIABLE' nodes. The first variable node stores the input as 'first\_name' (Type: STRING, Value: 'First Name'). The second variable node stores an exclamation mark '!' (Type: STRING, Value: '!').
- Terminal Output:** A terminal window titled 'Terminal' showing the program's execution. It displays the prompt 'What is your name? brandon' and the response 'Hello there, brandon!'. The terminal also shows the program's completion status: 'Finished running the program', 'Score: 8/2', and a link to 'See differences'.
- Assessment Report:** A window titled 'Off View' showing a table of results. The table has columns for 'Status', 'Line #', 'Submitted', and 'Expected'. It lists two lines of code, both marked as 'X' (failed), with the submitted code 'What is your name? brandon' and the expected code 'Hello there, brandon!'.



# Screenshots

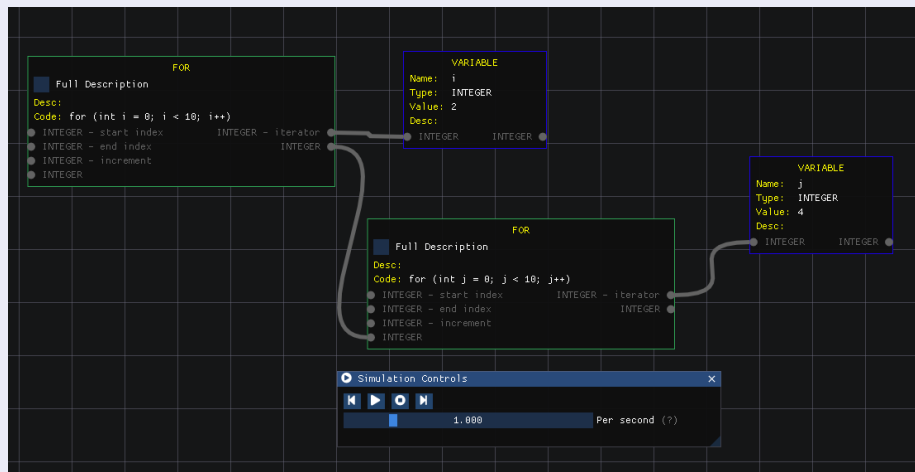
## Debug Module





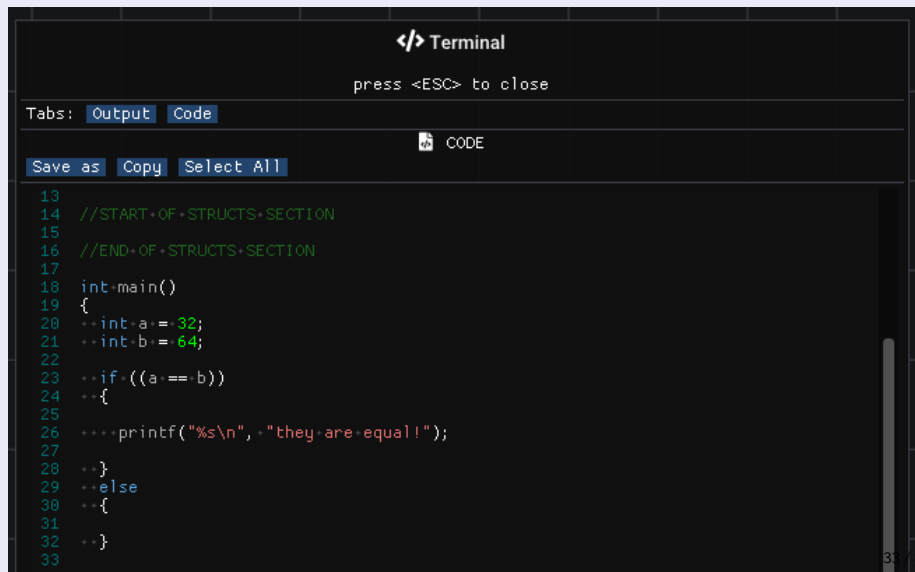
# Screenshots

## Simulation Module



# Screenshots


## Transpiler Module



**Terminal**

press <ESC> to close

Tabs: **Output** **Code**

 CODE

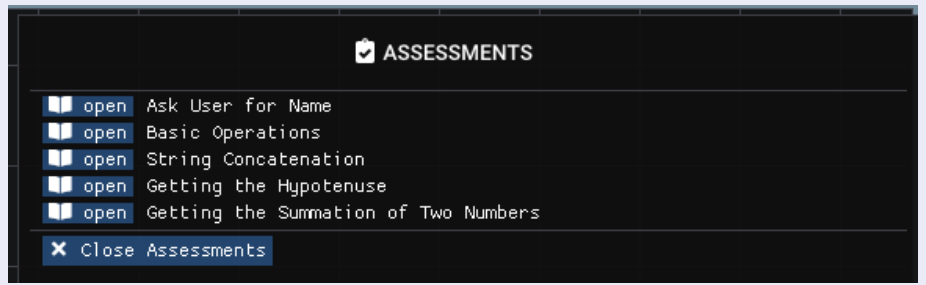
**Save as** **Copy** **Select All**

```
13
14 //START+OF+STRUCTS+SECTION
15
16 //END+OF+STRUCTS+SECTION
17
18 int+main()
19 {
20 ++int+a+=+32;
21 ++int+b+=+64;
22
23 ++if+((a+==+b))
24 ++{
25
26 ++printf("%s\n",+"they+are+equal!");
27
28 ++}
29 ++else
30 ++{
31
32 ++}
33
```



# Screenshots

## Assessments Module







# Screenshots

## Assessments Module

### Getting the Summation of Two Numbers

#### INSTRUCTION

1. Prompt the user for the **starting number**
2. Prompt the user for the **ending number**
3. Create a **variable** that will hold the sum with initial value of **0**
4. **Loop** using a **For** loop using the **starting** and **ending** numbers
5. **Add** the **sum variable** with the **iteration** count
6. After the loop, **print** the sum

#### EXPECTED OUTPUT

```
Enter starting number: 5
Enter ending number: 10
Summation: 45
```

[Do this assessment](#)[← Back](#)



# Screenshots

## Assessments Module

The screenshot displays the Assessments Module interface. On the left, a workflow diagram shows a sequence of steps: a **PROMPT** step, followed by a **VARIABLE** step, and then another **VARIABLE** step. The **PROMPT** step is highlighted with a yellow box. The **VARIABLE** steps are also highlighted with yellow boxes. The workflow is connected to a **Terminal** window in the center, which shows the output of the program. The terminal window has tabs for **Output** and **Code**. The **Output** tab is active, showing the text: "What is your name? brandon", "Hello there, brandon!", and "PRESS ENTER to EXIT". Below the terminal window, there is a **Diff Viewer** window showing a comparison of the actual output with the expected output. The **Diff Viewer** window has a table with columns for **Status**, **Line #**, **Submitted**, and **Expected**. The table shows two lines of output, both of which are marked as **OK** (green). On the right side of the interface, there is a panel with the following sections:

- INSTRUCTION**:
  1. Prompt the user for input for their answer
  2. Print a greeting + their name
- EXPECTED OUTPUT**:

What is your name? Brandon  
Hello there, Brandon!

[Cancel this assessment](#) [Back](#)



## Related Studies

- Prototype of Visual Programming Environment for C Language Novice Programmer (Abe, K., Fukawa, Y., & Tanaka, T., 2019)



## Related Studies

- Prototype of Visual Programming Environment for C Language Novice Programmer (Abe, K., Fukawa, Y., & Tanaka, T., 2019)
- On the Design of a Generic Visual Programming Environment (Zhang, D.-Q., & Zhang, K.)



## Related Studies

- Prototype of Visual Programming Environment for C Language Novice Programmer (Abe, K., Fukawa, Y., & Tanaka, T., 2019)
- On the Design of a Generic Visual Programming Environment (Zhang, D.-Q., & Zhang, K.)
- Environment pi J for Visual Programming in Java (Prokhorov, V., & Kosarev, V., 1999)



## Related Studies

- Prototype of Visual Programming Environment for C Language Novice Programmer (Abe, K., Fukawa, Y., & Tanaka, T., 2019)
- On the Design of a Generic Visual Programming Environment (Zhang, D.-Q., & Zhang, K.)
- Environment pi J for Visual Programming in Java (Prokhorov, V., & Kosarev, V., 1999)
- HASKEU: An editor to support visual and textual programming in tandem (Alam, A., & Bush, V. , 2016)



## Related Studies

- Prototype of Visual Programming Environment for C Language Novice Programmer (Abe, K., Fukawa, Y., & Tanaka, T., 2019)
- On the Design of a Generic Visual Programming Environment (Zhang, D.-Q., & Zhang, K.)
- Environment pi J for Visual Programming in Java (Prokhorov, V., & Kosarev, V., 1999)
- HASKEU: An editor to support visual and textual programming in tandem (Alam, A., & Bush, V. , 2016)
- The Scratch Programming Language and Environment (Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. ,2010)



## Related Studies

- Prototype of Visual Programming Environment for C Language Novice Programmer (Abe, K., Fukawa, Y., & Tanaka, T., 2019)
- On the Design of a Generic Visual Programming Environment (Zhang, D.-Q., & Zhang, K.)
- Environment pi J for Visual Programming in Java (Prokhorov, V., & Kosarev, V., 1999)
- HASKEU: An editor to support visual and textual programming in tandem (Alam, A., & Bush, V. , 2016)
- The Scratch Programming Language and Environment (Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. ,2010)
- CodeMonkey (Israel-Fishelson & HersHKovitz, 2020)





# Methodology - Materials

## Software Requirements - Development

- Linux 5.4 kernel with Manjaro distribution as Operating System



# Methodology - Materials

## Software Requirements - Development

- Linux 5.4 kernel with Manjaro distribution as Operating System
- Terminal for running commands



# Methodology - Materials

## Software Requirements - Development

- Linux 5.4 kernel with Manjaro distribution as Operating System
- Terminal for running commands
- Vim for text and code editing



# Methodology - Materials

## Software Requirements - Development

- Linux 5.4 kernel with Manjaro distribution as Operating System
- Terminal for running commands
- Vim for text and code editing
- GLFW and OpenGL for rendering



# Methodology - Materials

## Software Requirements - Development

- Linux 5.4 kernel with Manjaro distribution as Operating System
- Terminal for running commands
- Vim for text and code editing
- GLFW and OpenGL for rendering
- DearImGui and ImNodes for user-interface base framework



# Methodology - Materials

## Software Requirements - Development

- Linux 5.4 kernel with Manjaro distribution as Operating System
- Terminal for running commands
- Vim for text and code editing
- GLFW and OpenGL for rendering
- DearImGui and ImNodes for user-interface base framework
- C++ programming language



# Methodology - Method

## V-Model

This model follows the relationships between each of the different phases in the life cycle of the development process, each with an associated testing phase.

The primary focus and purpose of this model is to improve the efficiency of development and to ensure the effectiveness of the software.



# Methodology - V-Model

## Phases of V-Model

- Requirements





# Methodology - V-Model

## Phases of V-Model

- Requirements
- System Design



# Methodology - V-Model

## Phases of V-Model

- Requirements
- System Design
- Architecture Design



# Methodology - V-Model

## Phases of V-Model

- Requirements
- System Design
- Architecture Design
- Module Design



# Methodology - V-Model

## Phases of V-Model

- Requirements
- System Design
- Architecture Design
- Module Design
- Implementation and Coding



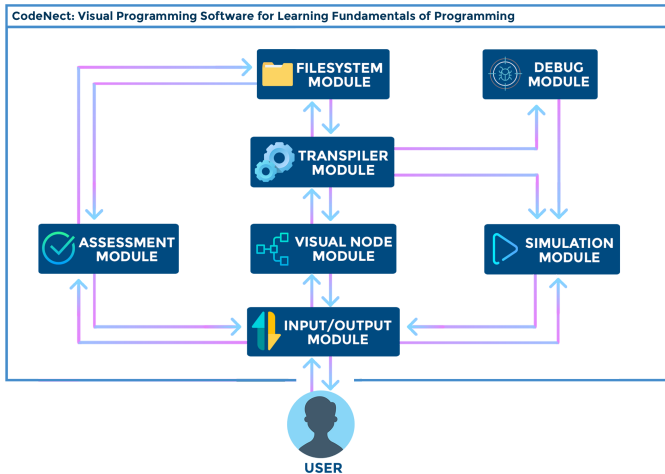
# Methodology - V-Model

## Phases of V-Model

- Requirements
- System Design
- Architecture Design
- Module Design
- Implementation and Coding
- Testings



# System Architecture





## Results and Discussion

### Likert Scale for Software Evaluation

LIKERT SCALE	
RANGE	INTEPRETATION
4.21 - 5.00	Excellent
3.41 - 4.20	Very Good
2.61 - 3.40	Good
1.81 - 2.60	Fair
1.00 - 1.80	Poor



# Technical Evaluation

## Profile

Name	E-Mail Address	Designation/ Rank	Institution	Educational Attainment
(not mentioned due to confidentiality)	auahdark687291@gmail.com	Software Engineer & Game Developer	Hasanuddin University	Information Systems
(not mentioned due to confidentiality)	thereal.alex.b@gmail.com	System Programmer & Lead Programmer	Syntacore	Computer Science
Fort Bautista	febhd0120@gmail.com	Web Developer	Snipesoft Ltd	Information Technology
Michael Gelvez	gelvezmichael@yahoo.com	Web Developer	Straight Login	Information Technology
Ronalyn De Guzman Rioflorida	ronrioflorida2@gmail.com	Web Developer & Trading Staff	Fatec Corporation	Information Technology
John Eros Puyo	johnerospuyo21@gmail.com	Instructor	Philippine Christian University	MIS
Ralph Waldo Candaza	rccandaza@up.edu.ph	Web Developer	Stratpoint	Computer Science





# Technical Evaluation

## Profile

Jaypee Galang	jaypeegalang27@gmail.com	Web Developer & Instructor	ISDC	Information Technology
Cyril Elijah Maurino	cyrilelijahaurino@gmail.com	Software Developer	Controtek Solutions	Computer Science
Conrad Reyes	conradreyes123@gmail.com	Web & Game Developer	Shopify	Information Technology



# Technical Evaluation

## Technical Evaluators

- 10 IT/CS professionals



# Technical Evaluation

## Technical Evaluators

- 10 IT/CS professionals
- Software downloaded from Google Drive



# Technical Evaluation

## Technical Evaluators

- 10 IT/CS professionals
- Software downloaded from Google Drive
- Software evaluated through Google Form



# Technical Evaluation

## Summary Table for the Overall of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
Functionality	4.33	0.82	excellent
Reliability	4.00	0.85	very good
Usability	4.10	0.83	very good
Efficiency	4.40	0.77	excellent
Maintainability	4.30	0.75	excellent
Portability	4.55	0.76	excellent
User-friendliness	4.43	0.67	excellent
<b>Average</b>	<b>4.30</b>	<b>0.78</b>	<b>excellent</b>



# Technical Evaluation

## Summary Table for the Functionality of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Informative (The information is clear, concise and informative to the intended audience.)	4.00	1.05	very good
2. Accurate (The software provides accurate and correct data.)	4.40	0.70	excellent
3. Interoperability (The modules are interconnected to each other and functions as a whole.)	4.60	0.70	excellent
<b>Average</b>	4.33	0.82	excellent



# Technical Evaluation

## Summary Table for the Reliability of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Reliable (The software is reliable in normal use.)	4.00	0.94	very good
2. Bug free (Software is bug free.)	3.79	0.95	good
3. Standard Equipment (The system uses standard equipment that is reliable, widely available and applicable to a variety of users.)	4.30	0.67	excellent
<b>Average</b>	4.00	0.85	very good



# Technical Evaluation

## Summary Table for the Usability of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Understandability (The software is easy to understand.)	4.00	1.05	very good
2. Operability (The software is easily operated by the intended user.)	4.20	0.63	very good
3. Learnability (The program is attractive and interesting; it motivates users to continue using the program.)	4.20	1.03	very good
<b>Average</b>	4.10	0.83	very good





# Technical Evaluation

## Summary Table for the Efficiency of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Special equipment (If the program requires special equipment, the requirements are minimal and clearly stated by the developer.)	4.40	0.70	excellent
2. Storage (The program doesn't consume large amount of memory that can slow down the processing of the system.)	4.40	0.84	excellent
3. Detection (The program can easily identify the cause of failure within the software.)	4.40	0.84	excellent
<b>Average</b>	4.40	0.77	excellent



# Technical Evaluation

## Summary Table for the Maintainability of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Function (The effort required to change the system functions is minimal.)	4.30	0.82	excellent
2. Process (The program is stable that if when something is changed, it will not affect the processing of the system.)	4.30	0.82	excellent
3. Test (The effort needed to test the system is minimal.)	4.30	0.67	excellent
<b>Average</b>	4.30	0.75	excellent



# Technical Evaluation

## Summary Table for the Portability of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Installation (The effort required to install the system is minimal.)	4.70	0.67	excellent
2. Adaptability (The system has the ability to adapt to new specifications or operating environments.)	4.40	0.84	excellent
<b>Average</b>	4.55	0.76	excellent



# Technical Evaluation

## Summary Table for the User-Friendliness of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Clarity of controls (Information about controls are understandable and available to the users.)	4.10	0.88	very good
2. Objectivity of contents (The language is non-discriminatory. Content is free from race, ethnic, gender, age and other stereotypes.)	4.90	0.32	excellent
3. Typographical Accuracy (The content is free from spelling and grammatical errors.)	4.30	0.82	excellent
<b>Average</b>	<b>4.43</b>	<b>0.67</b>	<b>excellent</b>



# Technical Evaluation

## Feedbacks

An interesting approach to teaching programming, I can see how assessments system can be used to teach data structures and algorithms in the future. Underlying C representation can also be instrumental in teaching proper memory management and even more complex concepts like cache. Overall software leaves a very positive first impression and a good extension opportunity.

Functionalities and features are working well. The software will be at best on its user-friendliness with sample demos at the menu.

Great Application!, I suggest it to have a version running on Linux based OS and in Mac OS also.



# Non-Technical Evaluation

## Profile

Name	E-Mail Address	Designation/ Rank	Institution	Course, Year and Section
Jerald Vidallo	jerald.vidallo @cvsu.edu.ph	Student	Cavite State University	BSIT 4-1
Ron Tseytlin	ronts390 @gmail.com	Student	Ben Gurion University	BSCS 1-1
Angelo Mari Paredes	angelomariparedes @gmail.com	Student	Luis Y. Ferrer Jr. Senior High School	STEM 2-1
Christian Vergel Plaus	mitoplaus @gmail.com	Student	De La Salle University	CPE 2-1
Edward Conception	edward.concepcion @cvsu.edu.ph	Student	Cavite State University	BSIT 4-1
Marie Joy Musa	mariejoy.musa @cvsu.edu.ph	Student	Cavite State University	BSIT 4-1
Marvin Recto	marvin.recto @cvsu.edu.ph	Student	Cavite State University	BSIT 4-1
Lmarl Saria	lmarlsaria21 @gmail.com	Graduate	Cavite State University	BSIT Graduate



# Non-Technical Evaluation

## Profile

Ren Antonio	reneantonio. dimabogte @cvsu.edu.ph	Student	Cavite State University	BSIT 2-4
Rhealyn Villar	rhealynvillar @gmail.com	Student	Cavite State University	BSIT 2-4
Jingkie Lagarde	jingkie.lagarde @cvsu.edu.ph	Student	Cavite State University	BSIT 3-3
Jaymark Abulencia	jaymark.abulencia @cvsu.edu.ph	Student	Cavite State University	BSIT 4-1



# Non-Technical Evaluation

## Non-Technical Evaluators

- 12 students with programming subjects





# Non-Technical Evaluation

## Non-Technical Evaluators

- 12 students with programming subjects
- Software downloaded from Google Drive



# Non-Technical Evaluation

## Non-Technical Evaluators

- 12 students with programming subjects
- Software downloaded from Google Drive
- Software evaluated through Google Form



# Non-Technical Evaluation

## Non-Technical Evaluators

- 12 students with programming subjects
- Software downloaded from Google Drive
- Software evaluated through Google Form
- YouTube video for tutorial on basic usage of the software



# Non-Technical Evaluation

## Summary Table for the Overall of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
Functionality	4.69	0.49	excellent
Reliability	4.41	0.67	excellent
Usability	4.37	0.73	excellent
User-friendliness	4.50	0.60	excellent
<b>Average</b>	4.50	0.62	excellent



# Non-Technical Evaluation

## Summary Table for the Functionality of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Informative (The information is clear, concise and informative to the intended audience.)	4.67	0.49	excellent
2. Accurate (The software provides accurate and correct data.)	4.83	0.39	excellent
3. Interoperability (The modules are interconnected to each other and functions as a whole.)	4.58	0.67	excellent
<b>Average</b>	4.69	0.52	excellent



# Non-Technical Evaluation

## Summary Table for the Reliability of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Reliable (The software is reliable in normal use.)	4.41	0.67	excellent



# Non-Technical Evaluation

## Summary Table for the Usability of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Understandability (The software is easy to understand.)	4.41	0.67	excellent
2. Learnability (The software is easily operated by the intended user.)	4.33	0.78	excellent
<b>Average</b>	4.37	0.73	excellent



# Non-Technical Evaluation

## Summary Table for the User-Friendliness of the Software

INDICATOR	MEAN	STANDARD DEVIATION	INTERPRETATION
1. Clarity of controls (Information about controls are understandable and available to the users.)	4.25	0.62	excellent
2. Objectivity of contents (The language is non-discriminatory. Content is free from race, ethnic, gender, age and other stereotypes.)	4.83	0.39	excellent
3. Typographical Accuracy (The content is free from spelling and grammatical errors.)	4.3	0.82	excellent
<b>Average</b>	4.50	0.60	excellent





# Non-Technical Evaluation

## Feedbacks

This is very useful for IT/CS students.

The interface of the program encouraged me to experiment and try out its different features.

The software needs to be more user-friendly. Without a manual, it is a little bit difficult to navigate and sometimes confusing. But the functionality of the software is built very well and the data shown are accurate.

Easy to understand just need time. better with instructions

Thank You and Its pleasure to be one of the first user of the app! Good day!

Excellent

Good program and file structure. It runs in Windows 10 w/o error.



# Summary

- Visual Programming Software for Learning Fundamentals of Programming



## Summary

- Visual Programming Software for Learning Fundamentals of Programming
- Uses visual elements instead of traditional text-based programming



## Summary

- Visual Programming Software for Learning Fundamentals of Programming
- Uses visual elements instead of traditional text-based programming
- Software usable by anyone with interest in learning programming



## Summary

- Visual Programming Software for Learning Fundamentals of Programming
- Uses visual elements instead of traditional text-based programming
- Software usable by anyone with interest in learning programming
- Supplementary software for instructors of programming



# Summary

## Seven Modules

- Visual Nodes Module



# Summary

## Seven Modules

- Visual Nodes Module
- Input/Output Module



# Summary

## Seven Modules

- Visual Nodes Module
- Input/Output Module
- Filesystem Module





# Summary

## Seven Modules

- Visual Nodes Module
- Input/Output Module
- Filesystem Module
- Transpiler Module



# Summary

## Seven Modules

- Visual Nodes Module
- Input/Output Module
- Filesystem Module
- Transpiler Module
- Debug Module



# Summary

## Seven Modules

- Visual Nodes Module
- Input/Output Module
- Filesystem Module
- Transpiler Module
- Debug Module
- Simulation Module



# Summary

## Seven Modules

- Visual Nodes Module
- Input/Output Module
- Filesystem Module
- Transpiler Module
- Debug Module
- Simulation Module
- Assessment Module



# Summary

## Evaluation

- 10 Technical Evaluators



# Summary

## Evaluation

- 10 Technical Evaluators
- 12 Non-Technical Evaluators



# Summary

## Evaluation

- 10 Technical Evaluators
- 12 Non-Technical Evaluators
- ISO 9126



# Summary

## Evaluation

- 10 Technical Evaluators
- 12 Non-Technical Evaluators
- ISO 9126
- Overall software evaluated as "EXCELLENT"





## Conclusion

- Seven modules developed and completed



## Conclusion

- Seven modules developed and completed
- Unit test and Integration test



## Conclusion

- Seven modules developed and completed
- Unit test and Integration test
- Technical evaluation overall mean = 4.30 (EXCELLENT)



## Conclusion

- Seven modules developed and completed
- Unit test and Integration test
- Technical evaluation overall mean = 4.30 (EXCELLENT)
- Technical evaluation overall mean = 4.50 (EXCELLENT)



## Conclusion

The lack of familiarity when it comes to visual programming has affected the metrics for usability for both the non-technical and technical evaluators as compared to traditional text-based programming, visual programming is rarely used or known.



# Unit Test

```
[doctest] doctest version is "2.4.6"
[doctest] run with "--help" for options
=====
../test.cpp:36:
TEST CASE: Testing Input/Output Module

0.000032 s: Testing Input/Output Module
=====
../test.cpp:59:
TEST CASE: Testing Filesystem Module

Opening invalid project

0.001556 s: Testing Filesystem Module
=====
../test.cpp:94:
TEST CASE: Testing Visual Nodes Module

Testing String Logic
To Array

0.063005 s: Testing Visual Nodes Module
=====
../test.cpp:727:
TEST CASE: Testing Transpiler Module

0.011802 s: Testing Transpiler Module
=====
../test.cpp:745:
TEST CASE: Testing Debugger Module

0.000227 s: Testing Debugger Module
=====
../test.cpp:773:
TEST CASE: Testing Simulation Module

Testing nested For-Loops
outer iteration #4
inner iteration #4

0.010103 s: Testing Simulation Module
=====
../test.cpp:823:
TEST CASE: Testing Assessments Module

2.820001 s: Testing Assessments Module
=====
[doctest] test cases: 7 | 7 passed | 0 failed | 0 skipped
[doctest] assertions: 558 | 558 passed | 0 failed |
[doctest] Status: SUCCESS!
```



## Recommendation

- Add more programming language target for transpilation. e.g Java



## Recommendation

- Add more programming language target for transpilation. e.g Java
- Increase the number of available nodes that can be used





## Recommendation

- Add more programming language target for transpilation. e.g Java
- Increase the number of available nodes that can be used
- Expand the number of assessment exercises



## Recommendation

- Add more programming language target for transpilation. e.g Java
- Increase the number of available nodes that can be used
- Expand the number of assessment exercises
- Expand the number of documents for guidelines and solutions



## Recommendation

- Add more programming language target for transpilation. e.g Java
- Increase the number of available nodes that can be used
- Expand the number of assessment exercises
- Expand the number of documents for guidelines and solutions
- In-depth tutorial/demo for absolute beginners



Thank you so much and God bless all!