International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

Principles of Programming : A Hypothetical Approach

SK.Shama*[1], M.Rithvik[2], D. Madhusudana Rao[3], K. Mohan Phani Kumar[4]
*[1]Assistant Professor, SRK Institute of Technology, Vijayawada, Andhra Pradesh, India
[2]Assistant Professor, SRK Institute of Technology, Vijayawada, Andhra Pradesh, India
[3]Assistant Professor, SRK Institute of Technology, Vijayawada, Andhra Pradesh, India
[4]B.Tech 2nd year,Student, SRK Institute of Technology, Vijayawada, Andhra Pradesh, India

ABSTRACT

In this paper we are going to introduce about the evaluation of programming languages. In recent days, E-learning is going to play a key role in the engineering education. In this scenario few subjects like Flat ,Compiler Design etc are developed in the form of VR application. So that a lay man can easily understand typical subjects. In this paper we are going to present the concept evaluation of programming languages in 2D view that resembles in 3D view.

Keywords : Virtual Reality (VR) application, Evaluation, 2D, 3D

I. INTRODUCTION

Computer play many important role in the society, helping to promote communication and interaction with the others and providing users a way to shop play games and have to access education.

we live in a fast-moving world where almost everything must come instantly to us. In this computer era, we depend on the computer to help us complete tasks, and to solve problems.

Significance of Computer programming:

Computers are used in various fields like business, pharmacy, music, education, engineering, defense, transportation, and cooking as they help to ease certain tasks, provide information faster, and speed.

A language is the main medium of communicating between the Computer systems and the most common are the programming languages. Computer only understands binary numbers that is 0 and 1 to perform various operations but the languages are developed for different types of work on a Computer. A language consists of all the instructions to make a request to the system for processing a task.

Principle of programming languages:

Programming is the process of coding, testing, troubleshooting, debugging and maintaining a system. Programming principles help you to write excellent quality of code and maintain a good coding practice

Program Efficiency Principles

1. **Readability**
   All developers were in a team able to understand the code.
2. **Extensibility**
   In the software world, we should not blame the change requests; it'll come at any time. So our code is always easy to extend without breaking existing business rules.
3. **Maintainability**
   It's easy to maintain by the development team and the production support team too because the application is loosely coupled.

4. **Modularity**
   Divide a program into reusable pieces: functions, modules, libraries

Types of Programming Principles

1.DRY(Don't-Repeat-Yourself)

Duplication can lead to maintenance nightmares, poor factoring, and logical contradictions. "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system". In other words, a piece of logic should only be represented once in an application.

- ✓ Duplication is the root of all software evils.
- ✓ DRY is also known as Duplication is Evil (DIE) or Once And Only Once.

2.KISS(KeepItSimple,Stupid)

Now a days programming languages frameworks , and APIs have powerful means to create sophisticated solutions for various kinds of problems. Sometimes developers might feel tempted to write "clever" solutions that use all these complex features.

The KISS principle states that most systems work best if they are kept simple rather than making them complex; therefore simplicity should be a key goal in design and unnecessary complexity should be avoided.

This principle can be applied to any scenario, including many business activities, such as planning, designing, and development.

3.YAGNI(YouAren'tGonnaNeedIt)

As developers, we'll always think a lot about the future usability of the project and try to do some extra features coding in a mind that "just in case we need them" or "we will eventually need them". Just one word… Wrong! I'll repeat it this way: You didn't need it, you don't need it, and in most of the cases… There are two main reasons to practice YAGNI

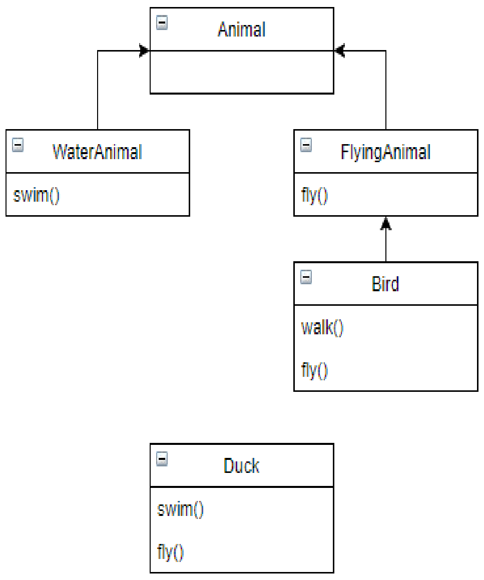
- ✓ You save time because you avoid writing code that you turn out not to need.
- ✓ Your code is better because you avoid polluting it with 'guesses' that turn out to be more or less wrong but stick around anyway.

4.SOLID

SOLID principle supports good object-oriented design and programming. Five of these principles are described as SOLID: Single responsibility, Open-closed, Liskov substitution, Interface segregation, and Dependency inversion.

5.Composition > Inheritance

The **"composition over inheritance" principle** states that objects with complex behaviors should do so by containing instances of objects with individual behaviors rather than inheriting a class and adding new behaviors.Overreliance on inheritance can lead to two major issues. First, the inheritance hierarchy can become messy in the blink of an eye. Second, you have less flexibility for defining special-case behaviors, particularly when you want to implement behavior from one inheritance branch in another inheritance branch:



Composition is a lot cleaner to write, easier to maintain, and allows for near-infinite flexibility as far as what kinds of behaviors you can define. Each individual behavior is its own class, and you create complex behaviors by combining individual behaviors.

6. Single Responsibility

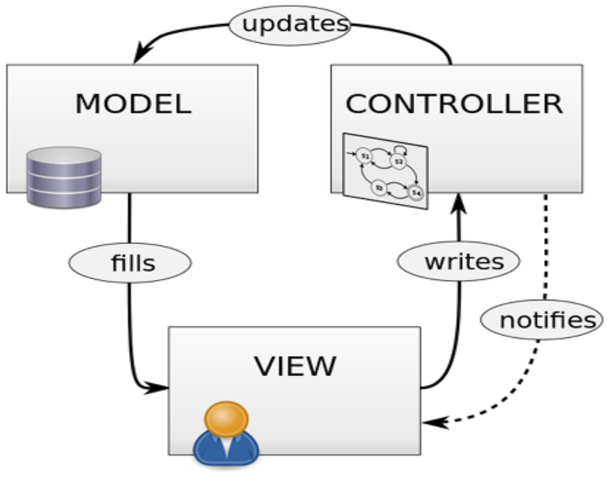
The **single responsibility principle** says that every class or module in a program should only concern itself with providing one bit of specific functionality. As Robert C. Martin puts it, “A class should have only one reason to change.”

Classes and modules often start off this way, but as you add features and new behaviors, it’s easy for them to evolve into God classes and God modules that take up hundreds, or even thousands, of lines of code. At this point, you should break them up into smaller classes and modules.

7. Separation of Concerns

The **separation of concerns principle** is like the single responsibility principle but on a more abstract level. In essence, a program should be designed so that it has many different non-overlapping encapsulations, and these encapsulations shouldn’t know about each other.

A well-known example of this is the model-view-controller (MVC) paradigm, which separates a program into three distinct areas: the data (“model”), the logic (“controller”), and what the end user sees (“view”). Variations of MVC are common in today’s most popular web frameworks.



For example, the code that handles the loading and saving of data to a database doesn’t need to know how to render that data on the web. The rendering code may take input from the end user, but then passes that input to the logic code for processing. Each part handles itself.This results in modular code, which makes maintenance much easier. And in the future, if you ever need to rewrite all of the rendering code, you can do so without worrying about how the data gets saved or the logic gets processed.

8.Avoid Premature Optimization

The **no premature optimization principle** is similar to the YAGNI principle. The difference is that YAGNI addresses the tendency to implement behaviors before they’re necessary while this principle addresses the tendency to speed up algorithms before it’s necessary.

The problem with premature optimization is that you can never really know where a program’s bottlenecks will be until after the fact.

9.Refactor, Refactor, Refactor

One of the hardest truths to accept as an inexperienced programmer is that **code rarely comes out right the first time**. It may feel right when you implement that shiny new feature, but as your program grows in complexity, future features may be hindered by how you wrote that early one.

Codebases are constantly evolving. It’s completely normal to revisit, rewrite, or even redesign entire chunks of code — and not just normal, but healthy to do so. You know more about your project’s needs now than when you did at the start, and you should regularly use this newly gained knowledge to refactor old code.

10.Clean Code > Clever Code

Speaking of clean code, leave your ego at the door and **forget about writing clever code**. You know what I'm talking about: the kind of code that looks more like a riddle than a solution and exists solely to show off how smart you are. The truth is, nobody really cares.

One example of clever code is packing as much logic into one line as possible. Another example is exploiting a language's intricacies to write strange but functional statements. Anything that might cause someone to say "Wait, what?" when poring over your code

II. METHODS AND MATERIAL

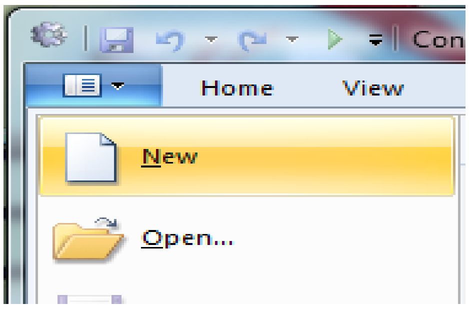
The application of learning the principles starts with the evolution of programming languages is designed with the help of a tool called construct2d

Installing Construct 2

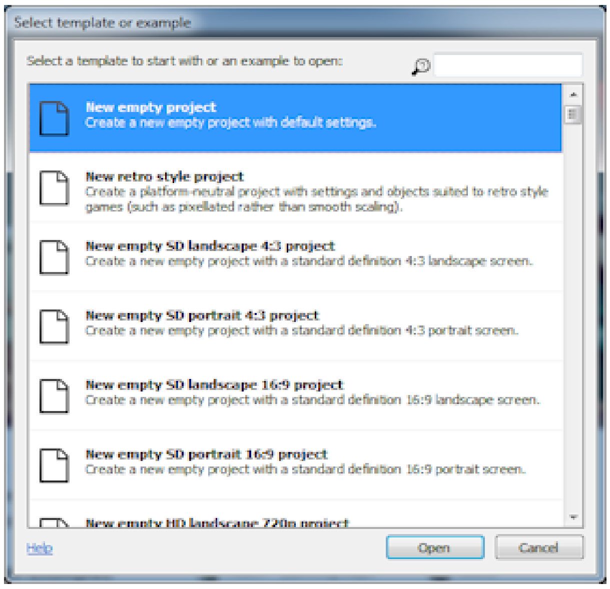
If you haven't already, grab a copy of the latest release of Construct 2.The Construct 2 editor is for Windows only, but the games you make can run anywhere, such as Mac, Linux or iPad. Construct 2 can also be installed on limited user accounts. It's also portable,

Getting started

Now you're set up, launch Construct 2. Click the File button, and select New.



You will see the 'Template or Example' dialog box.


Select template or example

Select a template to start with or an example to open:

New empty project
Create a new empty project with default settings.

New retro style project
Create a platform-neutral project with settings and objects suited to retro style games (such as pixellated rather than smooth scaling).

New empty SD landscape 4:3 project
Create a new empty project with a standard definition 4:3 landscape screen.

New empty SD portrait 4:3 project
Create a new empty project with a standard definition 4:3 portrait screen.

New empty SD landscape 16:9 project
Create a new empty project with a standard definition 16:9 landscape screen.

New empty SD portrait 16:9 project
Create a new empty project with a standard definition 16:9 portrait screen.

New empty HD landscape 720p project

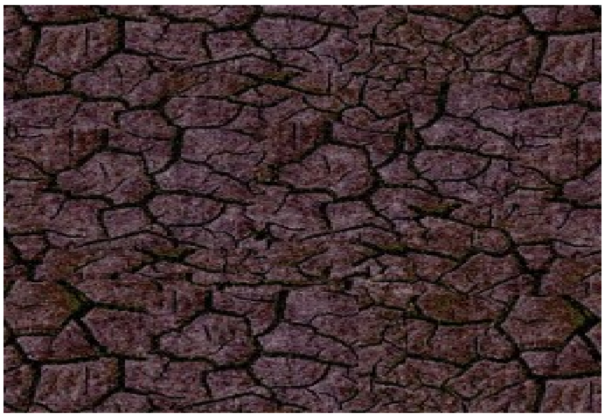
This shows a list of examples and templates that you can investigate at your leisure. For now, just click on 'Open' at the bottom of the box to create a blank, empty new project. Construct 2 will keep the entire project in a single .capx file for us. You should now be looking at an empty layout - the design view where you create and position objects.

Think of a layout like a game level or menu screen. In other tools, this might have been called a room, scene or frame.

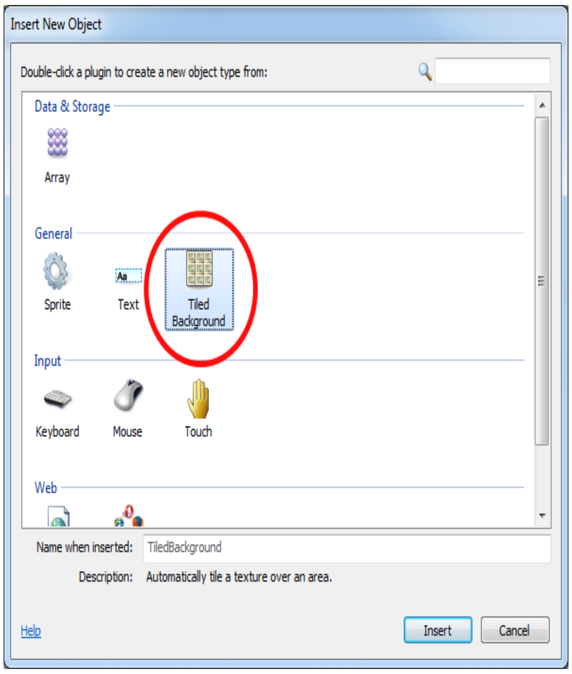
Inserting objects

1) Tiled Background

The first thing we want is a repeating background tile. The Tiled Backgroundobject can do this for us. First, here's your background texture - right click it and save it to your computer somewhere:



Now, **double click** a space in the layout to insert a new object. (Later, if it's full, you can also right-click and select Insert new object.) Once the Insert new objectdialog appears, **double click the Tiled Background object** to insert it.



Name when inserted: TiledBackground

Description: Automatically tile a texture over an area.

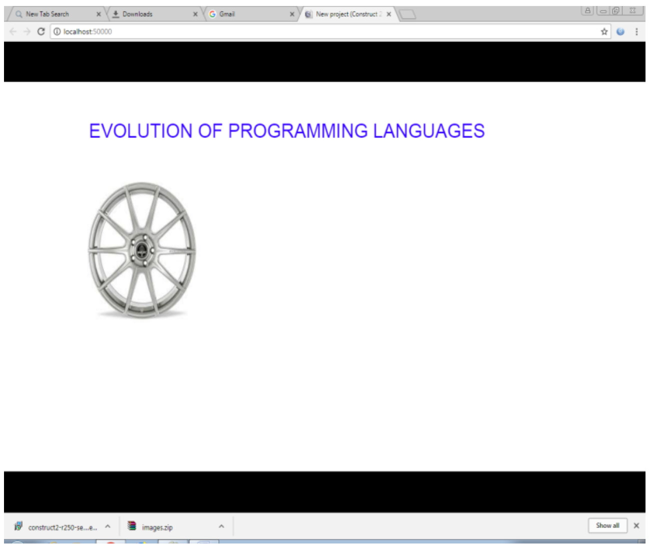
III. RESULTS AND DISCUSSION

This is a sort of E-Learning methodology where we compare evolution of programming languages with the evolution of vehicles.

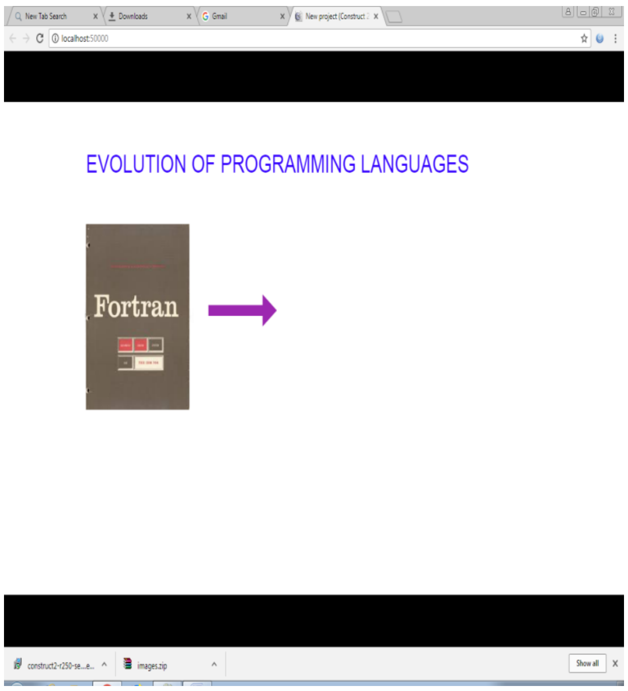
This type of comparison helps for a lay man to understand the concept of programming.

The below screen shot explains about the evolution of vehicles.

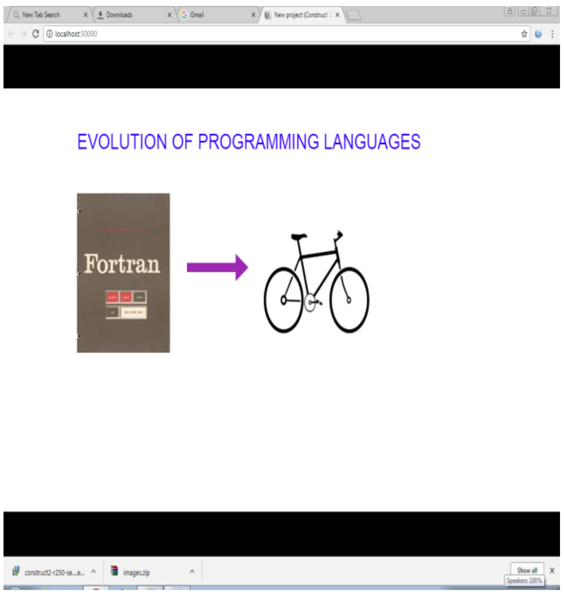
The application starts with the above text evolution of programming languages whenever the user clicks on the evolution of programming languages a wheel comes first.



When the user click on the wheel the first language that is evolved is displayed and thereby the user can understand the characteristics of first language

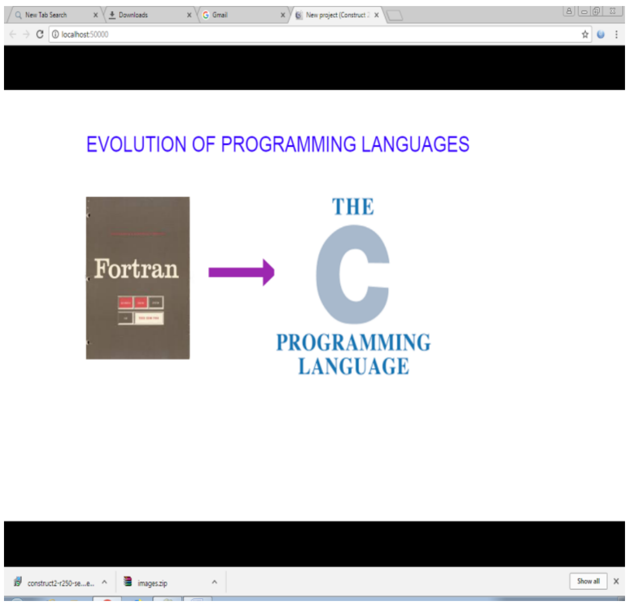


And the below screen shot explains the second generation of programming language by cliking the arrow 2nd generation of wheel that is cycle is displayed thus we can understand 2nd generation of programmming language



The user by clicking the cycle 2nd generation programming language is evolved that is the C prgramming language.

The user using this application can easily identify the programming languages evolution and characteristics easily



IV. CONCLUSION

This paper is a base to evaluate new era of E-Learning methodologies. The imagination of the user is coming in the form of technology in recent days in 3 dimensional forms called virtual reality.

By applying the same code by using the virtual reality applications will be a much better approach for the understanding of the student.

V. REFERENCES

[1]. Construct 2 tutorial

[2]. Principles of Programming Languages- Matteo Pradella Five Principles for Programming Languages for Learners By Mark Guzdial

VI. BIBILOGRAPHY

SK SHAMA

- ✓ **M.Tech.** (**Master of Technology**) in **CSE** with **9.3(cgpa)** from **Acharya Nagarjuna University of technology**, in 2014.
- ✓ **B.Tech.** (**Bachelor of Technology**) in **CSE** with **76.0%** from **Vasireddy Venkatadri Institute of Technology College** affiliated to Jawaharlal Nehru Technological University, Kakinada (JNTU Kakinada) in 2012.
- ✓ **Intermediate,** in Mathematics, Physics, Chemistry (**M.P.C**) with an aggregate of **94%** in Sri Chaitanya Junior college Guntur **in** 2008.
- ✓ **S.S.C.** with an aggregate of **93%** in Z.P.H School in 2006.
- ✓ Six months worked as Assistant Professor in St.Mary&'s Women's Engineering College.
- ✓ Worked as **Assistant Professor in Information Technology Dept.** at, RVR&JCCE Guntur, Andhra Pradesh since from 2015 to July 2017.
- ✓ Currently working as Assistant professor in Computer Science and Engineering Department at SRK Institutional Technology Vijayawada, from July 2017 to till date.

M.RITHVIK

- ✓ Gold medalist in **M .Tech** in **IT Department**.
- ✓ Have presented a PAPER ON **E-LEARNING METHODOLOGIES IN ASAR CONFERENCE** that makes or represents whole school in a rural village **HADDUBANGI, SRIKAKULAM DISTRICT** to attain an **International Status.**
- ✓ **"A STUDENT's Vision is a teacher's mission is his quote"** that makes him to succeed a class in his teaching.

D.MADHUSUDANA RAO

- ✓ He is an Entrepreneurial self-starter and a strong communicator.
- ✓ Fast learner and innovator.
- ✓ Active and enthusiastic worker.
- ✓ Can work as a team as well as individually.
- ✓ Ability to deal with people diplomatically.
- ✓ Love to learn new things.

K.MOHAN PHANI KUMAR

- ✓ One of the finest student in the class who has an enthusiasm to learn new things from the faculty and implement them in life.
- ✓ Good Active listener in the class
- ✓ Regular student to college
- ✓ Realized himself when he commits a mistake.
- ✓ Enthusiasm in seeking knowledge towards the new discoveries and innovations