Journal of Computer Science & Computational Mathematics, *Volume 11, Issue 3, September 2021*
DOI: 10.20967/jcscm.2021.03.007

Enhancing Computational Thinking of Children with Dyslexia using Visual Programming Languages Model

Aleena Nadeem*, Joveria Rubaab, Rabia Qayyum, Prof. Tauseef Rana

Department of Computer Software Engineering, MCS, National University of Sciences and Technology Islamabad, Pakistan
*Corresponding author email: aleena-nadeem@hotmail.com

***Abstract*:** After Steve Jobs admitted having symptoms of dyslexia, the subject has been highlighted to address concerns of programmers suffering from dyslexia. This paper is a qualitative study mapping computational skills to visual programming language tools that aids patients with dyslexia to overcome their deficiencies while designing and solving problems. It provides a detailed taxonomy of computational skills and tools of Visual programming environment that corresponds and help with a particular disability of dyslexic programmers. This research intends to help to identify Visual programming tools that allows dyslexic patients to design solution of problems more efficiently. Later on it identifies features of visual programming that would enhance performance of dyslexic programmers. Resultantly it devises a conceptual model of a language environment which incorporates all the supportive features for dyslexic programmers. This is beneficial for the future expansions of Visual programming environment/constructs to be developed keeping dyslexic patients in mind.

***keywords*:** Computational thinking (CT), visual programming language (VPL), dyslexia.

1. Introduction

A great number of programmers find it difficult to use textual languages. They struggle more to program or design a solution. This specific difficulty is named as dyslexia in medical Sciences. These people lack specific computational skills regarding reading and comprehension which makes coding through textual programming languages, a challenging task. Researches state that using Visual construction languages improves the programing capability and CT of dyslexic patients as this does not require struggling with syntax, letters and words. So we intend to map the elements of existing visual construction languages that are helpful in developing various CT skills in dyslexic people. Keeping our motivation of helping dyslexic patients in mind, this paper aim to suggest a customized environment for easing out the skill of programming for such individuals. This is done by providing particular facilitative features in visual programming languages that are helpful to dyslexic people.

In one of the researches [1] carried out for dyslexia, it is stated that around 10.4% people suffer from this issue in the world and out of a selected group of programmers, 12.4% claimed to have dyslexia. In large and medium sized organizations, people work in teams and groups to design solutions. Also, due to rapidly changing technology, agile processes are commonly practiced in organizations. The dyslexic member of the team might find it challenging to match their abilities to the non-dyslexic programmer of the team. Visual programming paradigm is a suggested solution of this gap in literature. Visual construction languages improves the programing capability and CT of dyslexic patients as this does not require struggling with syntax, letters and words. Figure 1 depicts conceptual model for the mapping of relationships between programming, CT, Dyslexia and Visual Programming that has been presented in paper.

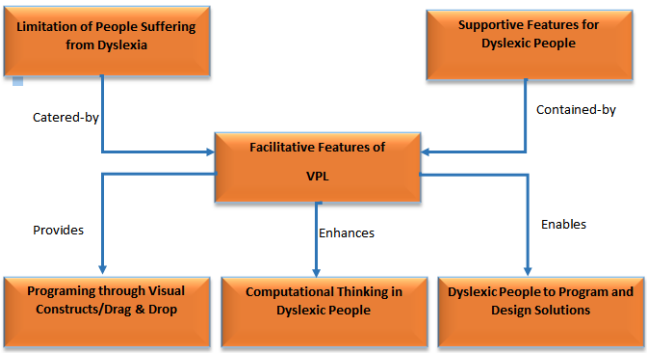

Figure 1. Mapping of relationship between Computational Thinking, Dyslexia, Programming and VPL

Computational skills are not confined to computer scientists and programmers alone. In the existing age they are considered significant for solving problems and designing solutions [2].They are various ways to apply computational skill. Textual programming is one the most widely used tool for application of computational skills. However. The work done in literature focuses on enhancing programming skills of people with dyslexia via Visual Programming tools. Scratch, Kodu and Lightbox are extensively used for recognizing programming patterns of dyslexic coder. This approach focuses on implementation of pre-built designs.

This paper is qualitative research and we have scanned computational perspectives, practices and concepts required by programmers to effectively design digital solutions. Later it builds taxonomy of the CT skills i.e. testing and debugging and control structure, to their corresponding facilitative feature used in Visual programming. Thereafter, we have identify computational skills deficient in dyslexic patients. Future, we have identified facilitative features of visual programming languages that will help in application of computational skills identified formerly. It will give an insight to designers to Visual Programming tools, to keep in consideration this specific percentage of dyslexic individuals. These It will help in future to construct customized environments for easing out the skill of programming for such individuals.

2. Background

In this paper a diverse survey has been carried out, regarding CT and its numerous dimensions. It has also explored various tools regarding Visual programming. This paper discovers various issues related to people suffering from dyslexia. In the following sections the aforementioned subjects will be conferred in detail.

Programming involves a series of steps which encompasses problem solving, applying pertinent computational theories and strategic depiction of the actual world in abstract form of programming language. But ironically many studies have shown that many of the people face extensive technical hitches in learning to program [3] to overcome this problem of acquiring the challenging skillset of textual programming, various visual programming environments have been developed in order to simplify the process of learning to program. VPL has been declared a prospective mean of supporting learners to grab the concepts of problem solving through computational skills. VPL supports learning by use of various visual constructs such as symbols, icons and signs etc. The learner can be supported to program anything by drag and drop of visual constructs/ elements and communicate pertinent strategies to develop computer program in an operational and structural way [4], [5], [6]

As far as CT is concerned, it is considered to be a vital skill for future generations and many researchers have backed up the argument that CT is essential skill that one must possess while programming CT can be perceived as a mental or conceptual tool required to solve problems and structure complex task [7].

A research through regression analysis explored the part of visual programming in computational thought process, and verified that learners can be benefited from many parts of visual constructs to answer computational problems with mapping of visual elements for a particular thought process at hand. For example iterations can be represented by loops and loops can be specified by a visual construct and a number of iterations a process has to go through. Resultantly student will develop an understanding of increments and iterations and distinguish the assessment of repeatable implementation and evaluation of evolving a program. The results disclosed that learners practiced various programming activities that differed in possessing numerous facets of CT [8].

CT is a very conventional term that was coined first time in 2006 by Wing in her article regarding CT. She stated that the concept is applicable to a skill set that can be possessed by anybody who is keen to learn and it is not necessarily be only possessed by computer scientist. Moreover (ISTE) International Society for Technology in Education grades CT just as algorithmic thinking with data representations and automation tool which utilizes simulation. Furthermore (NRC) National Research Council endorses CT and mathematics to be one of the indispensable eight practices for engineering and scientific aspect. (NRC 2012)

One of the most top rated studies of Brennan and Resnick (2012) projected three main dimensions of CT: computational concepts, computational practices, and computational perspectives. Table 1 sum up the fundamental ideas on three main dimensions [9]. This comprises of computational concepts that is syntactic and semantic knowledge. To add into it computational practices work as strategic knowledge. Moreover Table 2 maps CT dimensions to corresponding CT skills.

Table 1. Computational thinking dimensions

Computational Thinking	**Description**	**Examples**
Concepts	The concepts designers engage with	Variables
Practices	Practices that are used by designers and programmers while practicing the concepts	Loops that are iterative and incremental
Perspectives	Understanding of oneself, one's relationship to others and the technological domain around one.	Remixing and reusing, Testing and debugging, Modularizing and Abstracting

3. Related Work

Dyslexia is a Greek word that has been derived from two words, impaired (dys) and word skill (lexical). So basically dyslexia is a disability that is related to 'reading disability' which can be further associated with problems in interpreting unacquainted words or pronounceable no verses without meaning, but it also concerns writing disability [10] so Dyslexia is defined as:

"a specific learning difficulty which affects the ability to recognize words fluently and/or accurately; causes problems with spelling, auditory short-term memory, phonic skills,

multi-tasking, remembering instructions, and organizational skills" [11].

Around 10% of the people are living with dyslexia and the person suffering from dyslexia is different in many ways from another person, and there is much deliberation surrounding dyslexia support and identification [12].

Computer programming is mainly a text-based solution to a problem, and it proves to be additionally perplexing to the dyslexic programmer along with other challenges like cognitive tests of software implementation. The problems faced by dyslexic programmers have been defined both in relation to learning to program as well as understand the problem [13], [14].

In another research Dixon identified, the most persistent traits among dyslexic programmers were recognized and they made fewer slip-ups when at work with a VPL. Furthermore, 33.4% had improved ability to develop and comprehend visual programs than individuals that did not have signs of dyslexia, thus approving the hypothesis stated in this study [14].

Similarly the Program learning capability can also be enhanced in dyslexic programmers through the use of VPL. Programming behaviour of dyslexic patients has been studied in a research [15]. The paper has identified problems dyslexic patients face in programming. It has classified 31 VPLs and selected 5 languages. An experiment was conducted to help determine the preferences, strengths and performance differences of a group of programmers with and without dyslexia. The participants used two different programming environments i.e textual and visual to develop basic programs. Statistical results percentages presented in the results of the experiment support the idea that computer programmers with dyslexia perform better and have better skills in the development and monitoring of computer programs in a visual environment.
Moreover for identifying advantages of using visual programming in Novice users a research has been carried out to compare Scratch Jr and Light bot as an experiment on young children to map computational skills with techniques of aforementioned tools [16].

Researchers have found that patients of dyslexia have impacted and hindered effectiveness for the development practice of software. The study verifies the extensive issues faced by dyslexic patients while using computer and practicing programming with graphic aids and visual are more effective at understanding and development of computer programs as compared to the programmers without dyslexia. Moreover already stated that 12.4% of a group of programmers claim that they are dyslexic [1]. So by the use of VPL the CT of such people can be improved.

Table 2. Mapping of CT dimensions with corresponding CT skills

CT Concepts	CT practices	CT perspectives
Loops	Abstracting and modularising	Expressing
Variables	Being incremental and iterative	Questioning
Boolean Logic	Reusing and Remixing	Connecting
Events/ triggers	Reading, interpreting and communicating code	User interaction
Parallelism concurrency	Multi-modal design	
Sequences / Algorithms	Predictive thinking	
Conditionals		
Operators		
Selection		
Input/output		

4. Our Work

4.1 Computational Thinking Framework

In order to map computational skills to facilitative features of visual programming languages Brennan and Resnick's (2012) framework. Their model is based on concepts of coding. The three aspects in their model are parallel to the basis of programming concepts like conditionals and loops [17]. Since the paper deals with programming concepts and practices for dyslexic patients this frame work has maximum relevance. Moreover Brennan and Resnick's (2012) framework provides wide range of coverage of CT skills.

Lechang and jajal nouri carried out study of 55 papers regarding CT and they also identified CT skills other than Brennan and Resnick's (2012) framework. Those skills are separately identified and categorised. Table 2 contains computational skills that we will later map to VPLs features.
Computational concepts Loops (simple loops, nested loops, conditional loops), Boolean logic, events, parallelism (parallelism type I concurrency ascends when numerous objects have been executed on scripts side by side, input/output.Parallelism type II concurrency occurs when a specific object executes twice or more in script parallely, [18], sequences/algorithm, conditionals, operators and selection. *Computational Practices* Being incremental and iterative, Abstracting and modularising [1], reusing and remixing [4], testing and debugging [8] and reading, interpreting and communicating code [13], multi-modal design [6] and predictive thinking. *Computational Perspectives* Expressing [5],questioning [1], user interaction and connecting [3].

4.2 Kodu Programming Language

Fuertes in [15] has identified five programming languages that provide facilitative features for programmers with

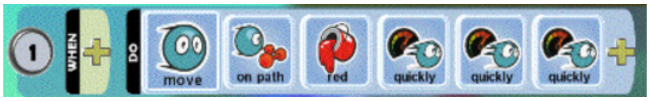

Figure 2. Object moving along a path

dyslexia. Kodu, Alice, App inventor, Snap and Scratch. This paper has analysed four the visual languages i.e. Kodu, Alice, App inventor and Scratch. Various facilitative features explored in Kodu are analyzed in the paper.

two modes of users in programming language provides two modes for working one is play mode and another is edit mode, both can be swapped, supporting iterative development. *Characters and Environmental elements* in Kodu provides programmable characters and environmental elements i.e. apples, rocks, coins and clouds. Environmental objects can be either public (movable objects) or private (immovable object). The degree of programmable depends on inherent attributes [19] *Paths* can be used to program objects to move in a direction or with constraints using paths. Figure 2 depicts how object can move along the red path. *Environments* in order for the coders can generate environment with terrain editing tools, World Tool also allows the programmer to begin world settings. It can utilized to establish glass walls, set camera mode, set wave height and strength and set day or night time etc. *Terrain Editors* allows coders to design terrain according to their wish. There are three terrain editing tools and a water tool provided to the programmer for designing anticipated terrain. *Conditions When and Do* in Kodu provide functionality for two conditional clauses, one is when and other is do. Figure 3 depicts the functionality in tiles GUI pattern, against a when clause a do tile can be added. *Sound Objects* for sound effects to trigger during events existences of the games. They can also be added as background music or with any other objects. *Boolean constructs of negation, indentation, and disjunction* because Boolean logic is difficult to grasp by students, however Kodu provides reasonably easier ways to implement this logic. *Negation or the not tile* is available in Kodu, for applying negation. Whenever not tile is used in junction with the tile negation is applied as in figure 4. *Indentation* in Kodu allows the user to indent lines of code which generates a logical dependence wherever the rules of the indented code are estimated only after the state of the parent instruction has been satisfied. *Disjunction* for logical disjunction is attained in Kodu by adding more than one rules with diverse circumstances but ensuing the similar act. *Graphical User Interface* provides all the options in a graphical form i.e. add, edit, erase, etc. Selector circle makes provision addition of objects to the world easier. *Context sensitive help* is available to help armature programmers. It provides valuable and applicable model of the tiles that could be used and the sequence they need to be placed in. It displays detailed narrative of the functionality these instructions will give.

The table 3 provides mapped CT skills and dimensions with the facilitative features of Kodu VPL.

Table 3. Mapping of facilitative characteristics of KODU to computational skills

VPL Facilitative feature	**Skill and Dimension**
3 Modes of users	Increment and iteration- CT practice
Characters and Environmental elements	Encapsulation,abstraction- CT practice
Sound Objects	Event trigger- CT concept
Paths	Predictive thinking- CT practice Expressing-CT perspective
Environments	Modularization- CT practice Expressing-CT perspective
Graphical User Interface	User Interaction- CT perspective
Terrain Editors	Multi-modal design- CT practice
Conditions When and Do	Conditional, trigger and event -CT concepts
Boolean constructs of negation, indentation, and disjunction	Operation, Sequence, Conditional-CT concept

Figure 3. Functionality of When-Do clause.

Figure 4. Visual representation of working of negation construct in Kodu

4.3 Scratch Programming Language

Scratch is block-based online programming language environment, designed for kids [20]. It practices a block based programming interface for designing game. Scratch has numerous features, enlisted in the paper.The features of the language and its corresponding CT skill and dimension is shown in table 4

Tinkerability allows user to touch any command or program fragment at any time to run the program or see what it can do. Visible execution can be observed via this feature. *Avoidance of Errors* is supported by shape of blocks acting as syntax. Since the blocks fit in a combination, as such in which statement of program is executed. The blocks do not fit in inappropriate ways, hence avoiding errors. *Function Blocks* act like operators, they can be used arguments for command block and they can be nested together to construct expressions. This chunk returns the value with respect to the function performed. *Trigger blocks* helps joining events to the stacks. *Object-based Programming language* such that scratch uses spirits, which encapsulates states- variables and behaviours-scripts. However, scratch does not have classes or inheritance, it's an object-based language but not an object-oriented programming languages. *Variables as concreate objects (Variable monitor)* such that variable are concrete objects i.e. banana, apple and pineapple are used as variable. *Minimal command blocks* by providning lesser lines of code depicts good functionality, therefore scratch allows programmers group similar functionality into one block, along with a drop down menu. Control blocks-Control blocks includes nesting, sequencing, repetition and iterations. *Data type* by providing few blocks with embedded parameter slots and the shape indicates the data type i.e. string, number, Boolean. Parameter with white background allows the user to enter a value, similarly parameter with can accept a function block. *Inter spirit communication and collaboration* because scratch uses a broadcast mechanism to support inter-sprite communication and synchronization. This enhances reusability and collaboration. It is an accommodating feature for provision of teamwork. *Command block* is provided so that commands can occur in a sequence like statements in textual programming language.

Table 4. Mapping of facilitative characteristics of SCRATCH to computational skills

VPL Facilitative feature	Skill and Dimension
Tinkerability	Modularization, abstraction, testing and debugging- CT Practice
Avoidance of Errors	Reading, interpreting and communicating code- CT Practice
Variables as concreate objects	Variable- CT concepts
Minimal command blocks	Abstraction- CT practice
Command block	Input output- CT concept Expressing- CT perspective
Function Blocks	Incremental and iterative- CT practice Selection- CT practice
Trigger blocks	Events/triggers- CT concept Conditionals–CT practice
Control blocks	Selection, loops, Boolean logic, sequences, selection- CT Concept
Data type	Sequences/algorithm, Operators- CT Concept
Object-based Programming language	Encapsulation, abstraction- CT practice
Inter spirit communication and Collaboration	Connecting- Perspective, Reusing and remixing- CT Practice Parallelism- CT Concepts
Trigger blocks	Events/triggers- CT concept Conditionals- CT practice

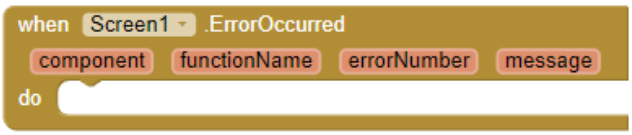

Figure 5. The trigger causes an event to perform some action

4.4 App Inventor Programming Language

MIT App Inventor is a natural, visual programming environment that allows everyone to build fully functional apps for smartphones and tablets [21]. The paper explores various GUI features of the App with respective CT skill and dimension in table 5.

*Restrictive Debugging*of App inventor's block language limits user from creating errors e.g. if a function block expects a number, the blocks editor restricts you from putting in a text field. *Block editor's* support programing of application behaviours using graphical displays. *Designer editor* helps in designing the UI of the application. *Computational Constructs* of the App Inventor allows basic computational constructs to be added in the code. It allows sequences, conditionals, loops, data structures to be added in the code. *Event- Driven programming* in App Inventor allows coders, design even-driven programming models. For instance, WHEN block can be used as an event trigger as "When text is Received". Do " ...." as depicted in figure 5. *Real Time testing* is also possible to track program execution on devices, including runtime errors. Whenever, user drags a new component or creates new functionality it is automatically made available to connected device or emulator. *Building Blocks* of App Inventor are common user interface elements i.e. buttons, labels, list pickers, images, sliders etc. These elements are coupled with mobile features like texting, GPS, NFC, wifi, Bluetooth etc. This allows coders add functionality in the application using user interface.

Table 5. Mapping of facilitative characteristics of APP INVENTOR to computational skills

VPL Facilitative feature	Skill and Dimension
Designer Mode	Predictive thinking, multimodal design- CT practice User Interaction- CT perspective
Block Mode	Expressing- CT perspective Sequences/algorithms- CT Concept
Building Blocks	User interaction- CT perspective
Computational Constructs	Conditionals- CT concepts
Event- Driven programming	Event/triggers-CT concept
Real Time testing	Testing debugging-CT practice Parallelism- CT concepts
Restrictive Debugging	Testing debugging- CT practice

Table 6. Mapping of facilitative characteristics of ALICE to computational skills

VPL Facilitative feature	Skill and Dimension
Camera is paned to a larger view	User Interaction- CT Perspective
Procedure and Functions	Expressing, Connecting- CT Perspective
Classes /Modularity	Modularity And Abstraction- CT Practice
Count up to Button	Loops- CT Practice
Do in Order	Loops- CT Practice
Flow of control	Sequence, Selection- CT Concept
Do together	Parallelism, concurrency- CT concept
Attributes of objects	Multimodal design- CT practice
Pre-defined methods	Abstraction, modularization, Reusing And Remixing- CT Practice
Arrays	Operators- CT Concept
Conditional functional	Conditionals-CT Concept

4.5 Alice Programming Language

Alice is an object-oriented programming language. The developers of Alice claim it to extremely helpful for novice programmers to leaver OOP [22]. *Count up to Button* provides for loop or while loop's functionality. *Do in Order* gives permit to the program to execute in sequence and execute all methods systematically. *Classes /Modularity* such that iconic classes are displayed, represented as pictorial folders and objects can be selected from these folders. Biped classes, flyer classes, prop classes are few of the examples as displayed in figure 6. *Attributes of objects* can be altered dynamically by means of default, rotation, translation and resize. *Predefined methods* can be incorporated in each class. They are present in resource section of the each class. They can be selected from a click of mouse. *Flow of control* of the language provides an opportunity to the programmer to have flow of control of the system. *Do together* feature allows the program to be executed concurrently, all the statements added in the block are executed consecutively. *Arrays* can help if we are using long animals as objects, which have tail and trunks like tiger or elephants, than arrays can be used to add joints. *Conditional functional* such that IF/Else, while, for all etc are present in the bottom of studio to be applied to the program. *Camera is paned to a larger view* so that the camera displays a larger view of the overall scene there by allowing users to visualize the entire story in one plane. *Procedure and Functions* for easy practice of functions and procedures a left pane is given to add and classify functions and procedures. Simple drag and drop functionality can be utilized to add these procedures and functions.
The table 6 discusses the supportive VPL feature of Alice and their mapping CT skills and dimensions.

Figure 6. Class selection from pictorial folders

4.6 Proposed Model

The facilitative supportive features of the four languages are selected by parallel analysis by use of languages and comprehensive study of the literature. Based on the analysis it is found that two languages Alice and Scratch are found to be better for dyslexic programmers as it covers most of the computational thinking skills and provide variety of features. The analysis also helped in pointing out few features that are present in one of the two languages and not well defined in the other. The solution thus proposed in the paper describes the model of a language that covers all of the missing features in these languages that are important for dyslexic patients in enhancing their performance. The model combines the features of Scratch and Alice along with few other features.
The features missing in Scratch based on comparison of two languages are; *Dynamical alteration of the objects and their attribute* which include size, rotation and translation of object to name the few. This feature helps support in the dimension of Computational Practice for enhancing Multimodal design as a skill in the dyslexic programmer. *Scratch is an object based language that provides that it support objects* e.g. Sprites but does not support object oriented paradigm of programming, it does not reach out to the concept of classes, their objects and inheritance. Thus

the dimension of computational practice will be improved enhancing better modularization and abstraction for the dyslexic programmer if it includes object oriented paradigm as well.

The features missing in Alice based on the comparison of two languages are; *The shape of blocks acts as syntax* and the blocks fit in a combination in a way in which statement of program is executed. The blocks do not fit in inappropriate ways, hence avoiding errors. This feature for avoiding errors lacks in Alice thus it lowers the skill of reading, interpreting and communication of code in the dimension of computational practice. *Alice programming language is not a good host for event/trigger relationship* that includes the do/when block providing to act on the certain statement when certain external or internal condition occurs. Inclusion of this feature enhances computational concept for the skill of event/trigger.

The features of Scratch and Alice that can work well for dyslexic people when combined are; *Scratch has an advantage of getting user accomplish a task in minimal code blocks* which makes it easier for user to interpret code and make better logic's for improving abstractions and modularization. This feature has an ability to enhance the computational practice of the dyslexic programmer. *Alice programming language is designed in a way that it provided good amount of pre-defined methods* that can be directly dragged in the pane and used in the program. These methods are designed in a generalized way that they can be reused in the code and fits in an appropriate way. These methods act as a module in which each module is assigned to complete a certain distinctive task. Computational practice is made available by making programmer able to achieve the skill set of good modularization and development by reuse.

Features that lack in both Scratch and Alice programming language are; There should be a feature that makes user understand the dragged command in a better way. Incorporating the changes that a dragged command can cause in the program shall be easily viewed before running the whole code or program. The user shall be able to pick good mental model of the command before actually running and testing the code. This enhances predictive thinking skill in order to increase the computational practices.

The above mentioned features of Scratch and Alice requires a new model of a language to be proposed that covers up the missing features of the two languages along with the combination of other important features as shown in conceptual model in figure 7. This model implicitly imports the features of Scratch and Alice. The proposed features are as follows:

Object oriented Programming and object based paradigm: Object oriented programming should be incorporated with object based programming. A certain class shall represent a certain object which has all its behavior and states. It makes information hiding easy and only required information is passed to any other object. For any similar behaviors,

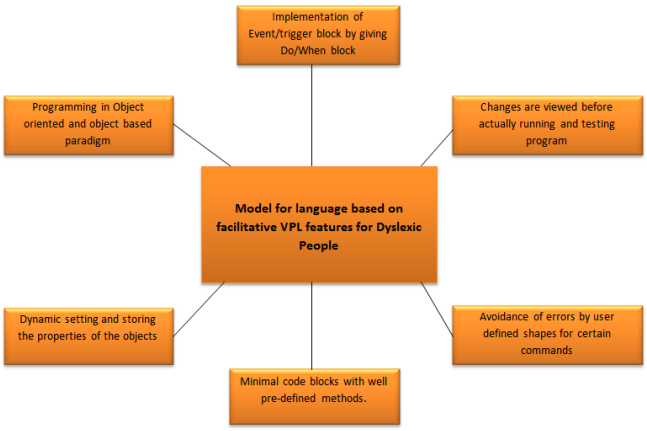

Figure 7. Conceptual model of proposed features for VPL environment

inheritance will provide better abstraction and handle good object communication. This makes objects more easy to implement. Computational Skill it enhances are Modularization and abstraction (CT practice), Encapsulation and expressing (CT perspective) and Reading, Interpreting and communicating code (CT practice)

Implementation of Event/trigger block by giving Do/When block: The language shall always provide with an event/trigger block namely trigger block which controls most of the internal and external conditions providing user with the facility of figuring out actions to perform when certain trigger occurs. The logic and control over the action makes user think more about the problem and less about how to implement in code. The trigger block may provide many pre-defined conditions that user is not able to think on its own. Computational skills it enhances are Event/triggers (CT Concept), Predictive Thinking (CT Practice) and Reusing and remixing of code (CT Practice)

Dynamic setting and storing the properties of the objects : The properties of the object or sprite shall be provided to be adjusted dynamically e.g. in case of the sprite, its size can be adjusted in the pane and stored in a variable by default. Computational skills it enhances is Multimodal design (CT Practice)

Avoidance of errors by user defined shapes for certain commands: The shapes shall be well defined for each certain type of command. The programmer shall be able to differentiate between different shapes for different command statements, controls and functions. The programmer should be given a support to define his own shapes for different statements for ease of use and understanding of code. This avoids syntax errors which encourages and motivates user to code well. Computational skills it enhances are Reading, interpreting and communicating code (CT Practice) and User interaction (CT Perspective)

Minimal code blocks with well pre-defined methods: By combining the features minimal code blocks and pre-defined methods, the program length can be minimized and easy to understand the code for dyslexic patients as most of the screen will not be occupied by the code blocks. The pre-

defined methods such as accessor and mutator method can be provided for programmer's convenience. Also predefined methods act as one module each which provides one functionality. Computational skills it enhances are Reading, Interpreting and communicating code (CT Practice), Reusing and remixing code (CT Practice) and Modularization and abstraction (CT Practice)

Changes are viewed before actually running and testing program: When a command is dragged from the command panel to the code pane, The user should be able to see what changes it incorporates to the object before running or testing the code. This makes good mental model of the command blocks and their functionalities which makes user know about what changes he is about to make. This allow user to make less effort in memorizing command functionalities and more effort on the logic of the code. Computational skills it enhances are Predictive thinking (Practice) and Questioning (Perspective)

5. Evaluation

The tool used for prototyping the language features is "Pencil". Pencil is a tool used for prototyping which is helpful in drawing free hand lines with hard edges. The tool is similar to paint but gives a free systematic approach to developers to begin drawing conceptual and physical models of the proposed model or solution. The mock-ups prepared by pencil help in building both a UI and a UX model.

In this paper, three features proposed in the model have been evaluated are minimal code blocks using predefined methods, use of event/trigger blocks for performing actions when an event occurs and different shapes to implement certain type of commands The evaluation of the proposed conceptual model of VPL environment is done by using the technique of prototyping the features by using the "Pencil". The results suggest that it is important to minimize the programmer's effort by reducing blocks and textual code (in blocks) on the screen as it makes the dyslexic programmer anxious to use the language. User's struggle for logical thinking regarding the actions to be performed and their corresponding events have been prototypes to be reduced. The results conclude that the manual effort for programming with respect to CT skills is minimal using the current conceptual model.

In figure 8 the prototype represents that the code blocks are kept minimal by using predefined get and set methods which are essential part of the object oriented programming language as the paper also includes that the OOP shall be combined with the object based programming. The set method is used to set the name of the object (sprite) in a string and get method is used to extract the string "I am Rachel Green". This example clearly states that the minimum code blocks are, the minimum effort the dyslexic people have to put without digging into what command to use to perform certain action. This is handled by the predefined methods.

The figure 9 prototypes that the events shall be already included as part of the language so that the dyslexic programmer has to only focus on defining the action without logical mental effort of thinking and including events. The screenshot displayed in the fig. is of the prototype made for concluding how an object shall respond to the message received where the user is already provided with options that if it's a message event that occurred or an error that shall be handled to program efficiently. In this example, the code block is used to explain that as soon as message "I am Racheal Green" the next object display a message "Hello Racheal Green" in the do block.

The two screens in the figure 8 and figure 9 shows that the commands are displayed in command pane following the consistency in the shapes of the objects. Different shapes are used for different type of commands but in case of similar set of commands same shape has been used. Here the prototype is not referring to the user defined shapes which the dyslexic programmer can define for help in the building blocks. The shapes represent clear differentiation for respective command and function blocks.

These three features are evaluated using the prototypes expressing the minimal area of screen used with textual code support in the blocks as they are specified as a built-in feature.

6. Conclusion

Dyslexic programmers cannot distinguish words, they have difficulty in recognising syntax of textual programming language, they lack at multi-tasking, they have trouble in memorising directions and they also find it problematic to work in a team [11]. However, visual programming languages provide multiple facilitative features for enhance computational skills of programmers.

The above mentioned tables give a diverse survey of facilitative features of Kodu, Scratch, App Inventor and Alice respectively. Kodu programming language permits programmers to design games, it gives varied features that supports multiple computational skills. App Inventor programming language is a VPL, for designing mobile or tablet application, however it does not support a huge variety of computational skills. Scratch and Alice on the other hand gives a huge variety of computational skills. Both the languages provide broad features for supporting and enhancing computational skills of dyslexic patients.

7. Future Work

The features in the proposed model are not implemented. The conceptual model in the paper has been evaluated using prototyping technique. The proposed model will be implemented as a programming environment for dyslexic programmers in the future. The respective research presented in the paper can be useful for the researchers who are interested in the area of introducing and implementing of new language which contains all the facilitative features for the shortcomings of dyslexic people. Moreover the research is helpful in designing and framing new teaching methodologies for people with dyslexia to enhance their computational abilities.

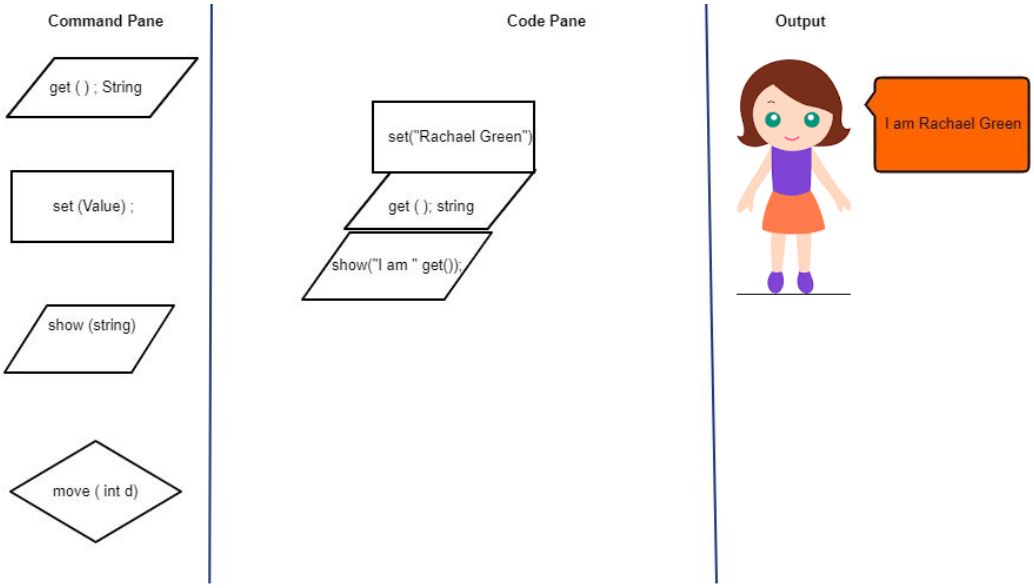

Figure 8. Prototype for proposed feature of minimal code blocks

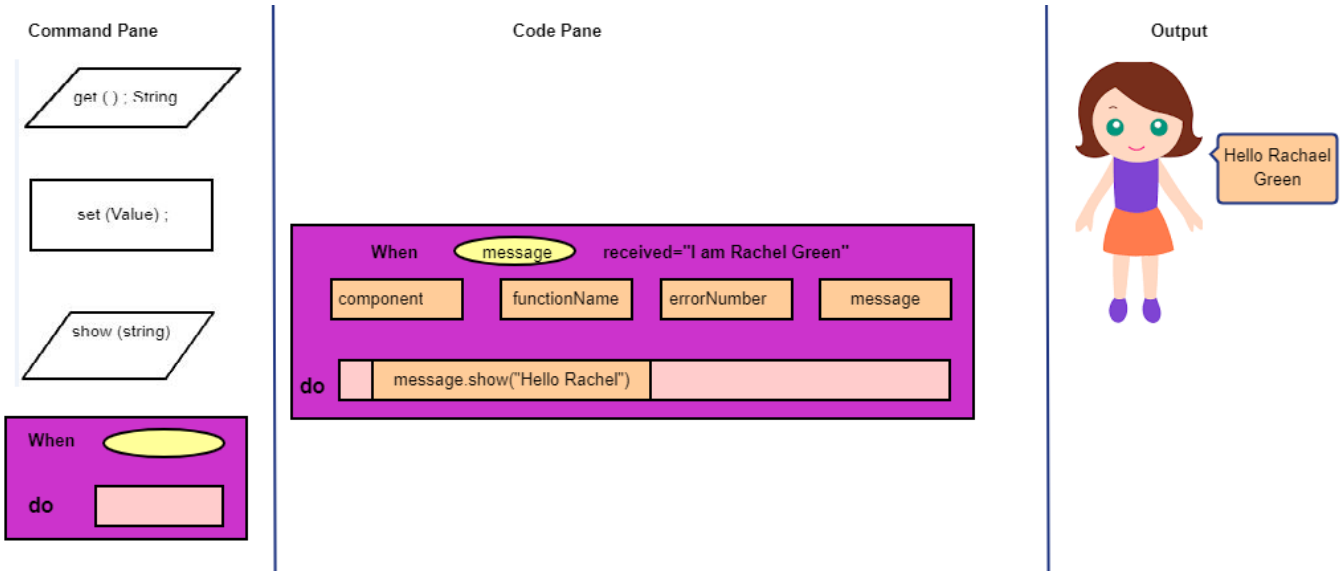

Figure 9. Prototype for proposed feature of event/trigger block

References

[1] J. L. Fuertes, L. F. González, and L. Martínez, "Characterization of programmers with dyslexia," in *International Conference on Computers Helping People with Special Needs*. Springer, 2016, pp. 339–342.

[2] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer science education*, vol. 13, no. 2, pp. 137–172, 2003.

[3] F. W. Li and C. Watson, "Game-based concept visualization for learning programming," in *Proceedings of the third international ACM workshop on Multimedia technologies for distance learning*. ACM, 2011, pp. 37–42.

[4] L. A. Gouws, K. Bradshaw, and P. Wentworth, "Computational thinking in educational activities: an evaluation of the educational game light-bot," in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. ACM, 2013, pp. 10–15.

[5] I. Paliokas, C. Arapidis, and M. Mpimpitsos, "Playlogo 3d: A 3d interactive video game for early programming education: Let logo be a game," in *2011 Third International Conference on Games and Virtual Worlds for Serious Applications*. IEEE, 2011, pp. 24–31.

[6] S. Xinogalos, "An evaluation of knowledge transfer from microworld programming to conventional programming," *Journal of Educational Computing Research*, vol. 47, no. 3, pp. 251–277, 2012.

[7] S. Cooper, W. Dann, and R. Pausch, "Alice: a 3-d tool for introductory programming concepts," *Journal of Computing Sciences in Colleges*, vol. 15, no. 5, pp. 107–116, 2000.

[8] C.-C. Lin, P.-Y. Chao, E.-T. Lin, and H.-L. Tzeng, "Exploring the role of visual programming activities in computational thinking," in *2018 1st International Cognitive Cities Conference (IC3)*. IEEE, 2018, pp. 135–138.

[9] L. Zhang and J. Nouri, "A systematic review of learning by computational thinking through scratch in k-9," *Computers & Education*, p. 103607, 2019.

[10] R. Thompson, S. Tanimoto, R. D. Lyman, K. Geselowitz, K. K. Begay, K. Nielsen, W. Nagy, R. Abbott, M. Raskind, and V. Berninger, "Effective instruction for persisting dyslexia in upper grades: Adding hope stories and computer coding to explicit literacy instruction," *Education and information technologies*, vol. 23, no. 3, pp. 1043–1068, 2018.

[11] S. Wallace, *A dictionary of education*. OUP Oxford, 2015.

[12] D. Armstrong and G. Squires, *Key Perspectives on Dyslexia: An essential text for educators*. Routledge, 2014.

[13] N. Powell, D. Moore, J. Gray, J. Finlay, and J. Reaney, "Dyslexia and learning computer programming," 2004.

[14] M. Dixon, "Comparative study of disabled vs. non-disabled evaluators in user-testing: dyslexia and first year students learning computer programming," in *International Conference on Universal Access in Human-Computer Interaction*. Springer, 2007, pp. 647–656.

[15] J. L. F. Castro, L. F. G. Alvarán, and L. A. M. Normand, "Visual programming languages for programmers with dyslexia: An experiment," in *2018 IEEE 14th International Conference on e-Science (e-Science)*. IEEE, 2018, pp. 145–155.

[16] S. Rose, J. Habgood, and T. Jay, "An exploration of the role of visual programming tools in the development of young children's computational thinking," *Electronic journal of e-learning*, vol. 15, no. 4, pp. 297–309, 2017.

[17] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, vol. 1, 2012, p. 25.

[18] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari, "Learning computer science concepts with scratch," *Computer Science Education*, vol. 23, no. 3, pp. 239–264, 2013.

[19] A. Fowler, T. Fristce, and M. MacLauren, "Kodu game lab: a programming environment," *The Computer Games Journal*, vol. 1, no. 1, pp. 17–28, 2012.

[20] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, p. 16, 2010.

[21] S. C. Pokress and J. J. D. Veiga, "Mit app inventor: Enabling personal mobile computing," *arXiv preprint arXiv:1310.2830*, 2013.

[22] J. Dwarika and M. R. De Villiers, "Use of the alice visual environment in teaching and learning object-oriented programming," in *Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists*. ACM, 2015, p. 14.