Game-based Learning to Teach Computational Thinking to High School Students.

Guided by

Prof. Ravi Poovaiah

IDC School of Design, IIT Bombay

By

Lokesh Kumar V M

(206330003)

M.Des Interaction Design (2020-2022)

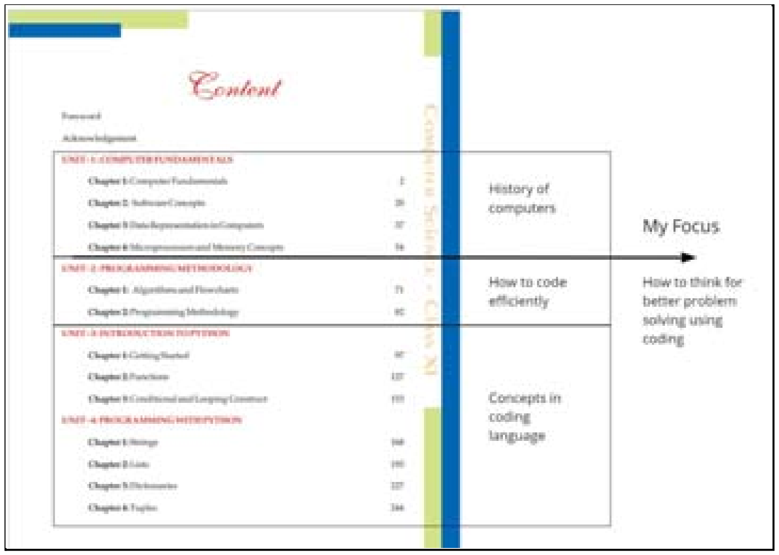
IDC School of Design, IIT Bombay

1. Introduction

Children must be taught how to think, not what to think -Margaret Mead.

All over the globe, governments were introducing some form of computing curriculum from kindergarten—to five years onwards. Over 30 countries have introduced a mandatory computer science curriculum for grades KG to 12. With the increase in global demand, the NEP 2020 of India has made coding a mandatory subject from class 6.
The government took all these initiatives to prepare students for 21st-century skills. Computation skills, thinking skills, innovation skills, problem-solving skills, and others were identified as essential parts of 21st-century skills. However, sadly only very few parts of the curriculum accommodate such a pedagogical approach to tackle this problem.

After analyzing the current structure of the CBSE curriculum, starting from the content of the 11th std CBSE computer science book since programming officially starts in the 11th grade, when students choose their respective stream. The index page of the book has been shown for reference in figure 1. There are no lessons on how to acquire the thinking skills required to code.



Content

Foreword

Acknowledgement

UNIT - 1: COMPUTER FUNDAMENTALS

Chapter 1: Computer Fundamentals	2
Chapter 2: Software Concepts	20
Chapter 3: Data Representation in Computers	37
Chapter 4: Microprocessors and Memory Concepts	54

UNIT - 2: PROGRAMMING METHODOLOGY

Chapter 1: Algorithms and Flowcharts	71
Chapter 2: Programming Methodology	85

UNIT - 3: INTRODUCTION TO PYTHON

Chapter 1: Getting Started	97
Chapter 2: Functions	127
Chapter 3: Conditional and Looping Construct	155

UNIT - 4: PROGRAMMING WITH PYTHON

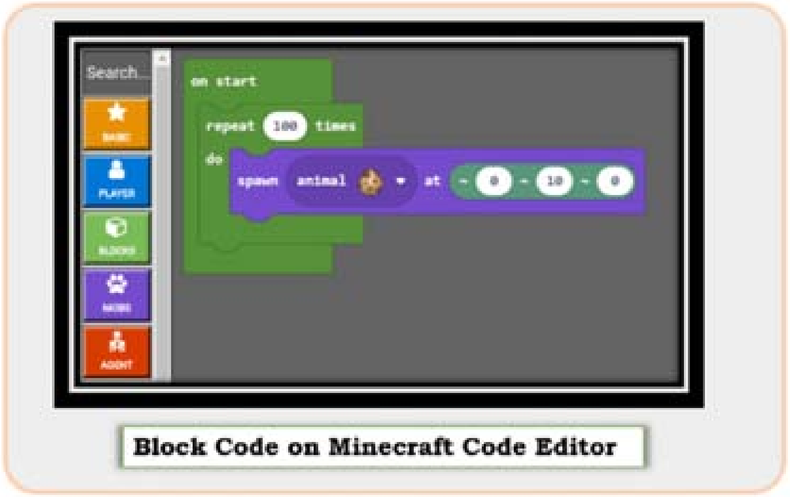
Chapter 1: Strings	168
Chapter 2: Lists	190
Chapter 3: Dictionaries	227
Chapter 4: Tuples	244

Computer Science - Class XI
History of computers
How to code efficiently
Concepts in coding language
My Focus
How to think for better problem solving using coding

Fig 1. 11th Std CBSE Computer science book.

However, in advance of, NEP 2020. CBSE has collaborated with Microsoft to develop resources that focus on project-based learning. Especially using visual coding language, the trend of using boring text-based coding language has been changing to visual coding
language in the initial stage of learning to code. NEP 2020 uses Minecraft code, make code as shown in figure 2 and 3, and scratch along with python to teach coding from the sixth to the eighth standard. These are introductory courses.

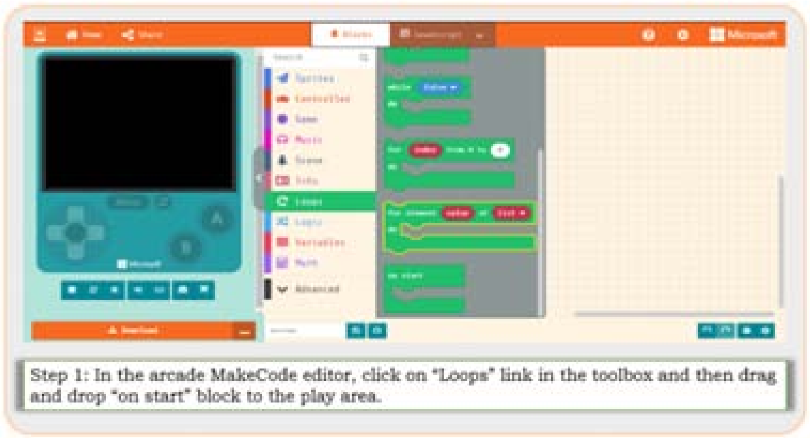
With an increase in the complexity of the curriculum, it requires an increase in pedagogical techniques and competencies. Especially in computer science, where coding will be introduced, in order to make it easy for students, a strong foundation is needed, which can be achieved through teaching how to think for such computations rather than teaching the coding language or various algorithms. We focus primarily on cognitive ability called "Computational thinking" and making it exciting and easy to understand through a game-based learning approach for students entering high school.



```
on start
  repeat 100 times
  do
    spawn animal ▾ at ~ 0 ~ 10 ~ 0
```

Block Code on Minecraft Code Editor

Fig 2. Minecraft Code by Microsoft, to teach coding.



Step 1: In the arcade MakeCode editor, click on "Loops" link in the toolbox and then drag and drop "on start" block to the play area.

Fig 3. Make code by Microsoft to teach coding.

2. Content

Why Computational thinking?

Computational Thinking could enable students to **think in a different** way while **solving problems**, analyze everyday issues from a different perspective, and even **apply it to various other subjects,** apart from computer science.

What is Computational Thinking?

Definitions of Computational thinking

various computer scientists across the world curate several definitions of computational thinking. Definitions of Computational thinking as given in the book "Developing Computational Thinking in Compulsory Education by the European Commission, Joint Research Centre" [9].

- *"Computational thinking is the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes".*

- *"Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent".*

- It is about Conceptualizing, not about programming.
- It is a fundamental skill, not a rote skill.
- A human way of thinking, not how the computer thinks.
- Complements and combines mathematical and engineering thinking.
- It is about Ideas, not artifacts.

The International Society for Technology in Education (ISTE), in collaboration with the Computer Science Teachers Association (CSTA) published a definitional list of computational thinking characteristics [6]. These include the point below but are not limited to:

1) Formulating problems for use with a computer to facilitate the solution.

2) Logically organizing & analyzing data.

3) Representing data through abstractions.

4) Automating solutions through algorithmic processes.

5) Identifying, analyzing, and implementing a possible solution as the most efficient & effective combination of steps and resources.

6) Generalizing and transferring this process to a variety of problem areas.

Comparison between Design thinking & computational thinking.

In this section, we talk about understanding computational thinking with respect to the design thinking process that designers generally aware of.

“Design thinking and computational thinking: a dual-process model for addressing design problems” [13] by Kelly, N., & Gero explains the relationship between design thinking and computational thinking and how to use them in the design process. The graph is created by two orthogonal ontological categories, with DT & CT located in space, as shown in figure 4.

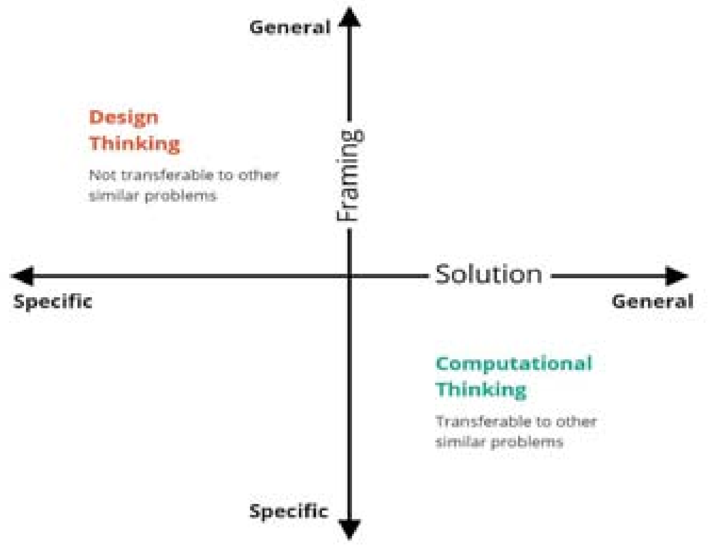
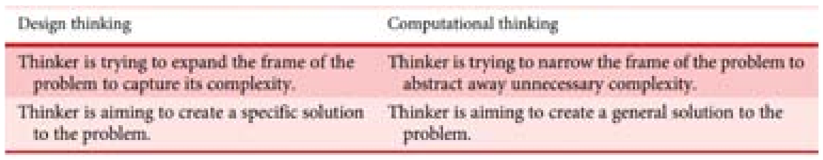


Fig 4. Graph showing the relation between DT and CT.

Table 1, shows the dual-process model where a designer used both thinking skills to solve the problem.



Design thinking	Computational thinking
Thinker is trying to expand the frame of the problem to capture its complexity.	Thinker is trying to narrow the frame of the problem to abstract away unnecessary complexity.
Thinker is aiming to create a specific solution to the problem.	Thinker is aiming to create a general solution to the problem.

Table 1. The dual-process model of design thinking and computational thinking.

As suggested in “Design thinking and computational thinking: a dual-process model for addressing design problems, " both processes were used in this project.

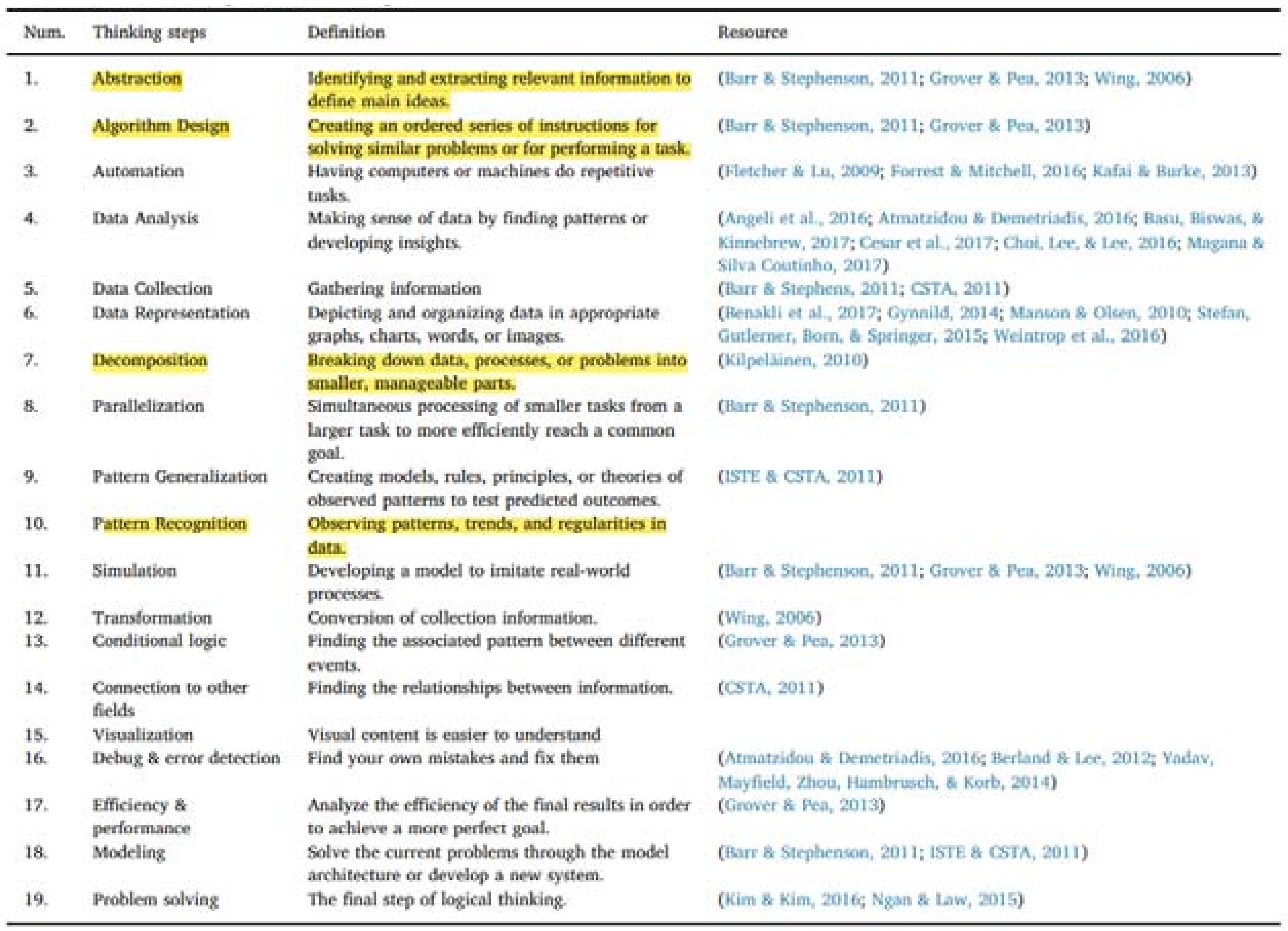
Using design thinking to identify the big picture (like understanding the user and defining problem space.) and computational thinking to figure out the sub-problems (like breaking down what needs to be taught and elements to consider in-game).

Importance of Computational thinking

- Computational thinking can also be applied in STEM and anywhere that requires a computer to solve a problem.
- It is a way of human thinking, it is not the computer's calculation model.
- It is used to extend the foundation of mathematics using a combination of mathematical thinking and engineering thinking.
- It is a way of the finished product of thinking.
- It addresses most of the 21st-century skills.
- Many developed countries have recognized the potential of Computational thinking and included it in the educational curriculum starting from kinder garden.

Concepts in Computational Thinking

There are various concepts in the computational thinking process [5] used in problem-solving, as mentioned in table 2.



Num.	Thinking steps	Definition	Resource
1.	Abstraction	Identifying and extracting relevant information to define main ideas.	(Barr & Stephenson, 2011; Grover & Pea, 2013; Wing, 2006)
2.	Algorithm Design	Creating an ordered series of instructions for solving similar problems or for performing a task.	(Barr & Stephenson, 2011; Grover & Pea, 2013)
3.	Automation	Having computers or machines do repetitive tasks.	(Fletcher & Lu, 2009; Forrest & Mitchell, 2016; Kafai & Burke, 2013)
4.	Data Analysis	Making sense of data by finding patterns or developing insights.	(Angeli et al., 2016; Atmatzidou & Demetriadis, 2016; Basu, Biswas, & Kinnebrew, 2017; Cesar et al., 2017; Choi, Lee, & Lee, 2016; Magana & Silva Coutinho, 2017)
5.	Data Collection	Gathering information	(Barr & Stephens, 2011; CSTA, 2011)
6.	Data Representation	Depicting and organizing data in appropriate graphs, charts, words, or images.	(Benakli et al., 2017; Gynnild, 2014; Manson & Olsen, 2010; Stefan, Gutlerner, Born, & Springer, 2015; Weintrop et al., 2016)
7.	Decomposition	Breaking down data, processes, or problems into smaller, manageable parts.	(Kilpeläinen, 2010)
8.	Parallelization	Simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal.	(Barr & Stephenson, 2011)
9.	Pattern Generalization	Creating models, rules, principles, or theories of observed patterns to test predicted outcomes.	(ISTE & CSTA, 2011)
10.	Pattern Recognition	Observing patterns, trends, and regularities in data.	
11.	Simulation	Developing a model to imitate real-world processes.	(Barr & Stephenson, 2011; Grover & Pea, 2013; Wing, 2006)
12.	Transformation	Conversion of collection information.	(Wing, 2006)
13.	Conditional logic	Finding the associated pattern between different events.	(Grover & Pea, 2013)
14.	Connection to other fields	Finding the relationships between information.	(CSTA, 2011)
15.	Visualization	Visual content is easier to understand	
16.	Debug & error detection	Find your own mistakes and fix them	(Atmatzidou & Demetriadis, 2016; Berland & Lee, 2012; Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014)
17.	Efficiency & performance	Analyze the efficiency of the final results in order to achieve a more perfect goal.	(Grover & Pea, 2013)
18.	Modeling	Solve the current problems through the model architecture or develop a new system.	(Barr & Stephenson, 2011; ISTE & CSTA, 2011)
19.	Problem solving	The final step of logical thinking.	(Kim & Kim, 2016; Ngan & Law, 2015)

Table 2. List of categories in Computational thinking.

Scope of project

Out of the 19 concepts in computational thinking, the highlighted ones are the most important and the bare minimum to achieve computational thinking.
For the scope of the project, we were focusing on the majorly accepted concepts in Computational thinking which will be enough for problem-solving and formulate proper instructions for the computer. Which are **Decomposition, Pattern Recognition, Abstraction, and Algorithm** as shown in figure 5**.** This project focuses on teaching it in a simple, fun and easy way to understand, without using computer science.
Also, the project doesn't focus on any kind of computer science concepts like coding languages or coding concepts like data structures, syntax, OOPS, etc.
A simple example of computational thinking is mentioned in figure 6.

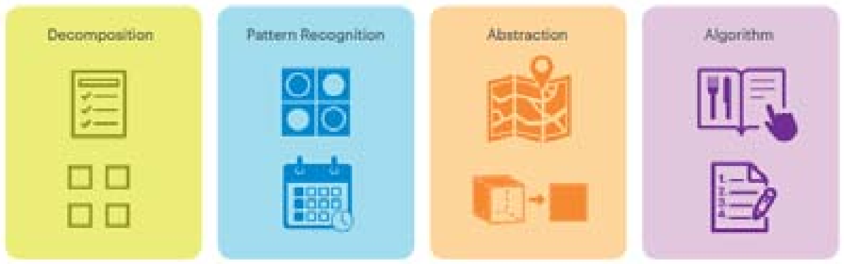


Fig 5. Concepts in computational thinking

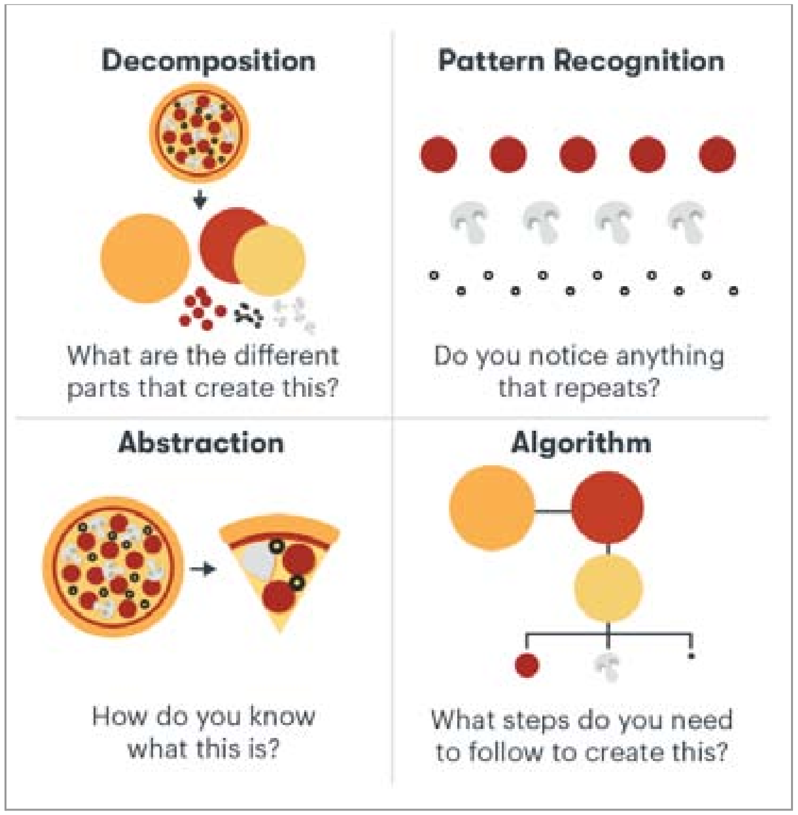

Decomposition
What are the different parts that create this?
Pattern Recognition
Do you notice anything that repeats?
Abstraction
How do you know what this is?
Algorithm
What steps do you need to follow to create this?

Fig 6. Example of computational thinking in making pizza.

The following section explains the four steps in computational thinking, (Decomposition, Pattern Recognition, Abstraction, and Algorithm).

Concepts in focus

1. Decomposition

Breaking down problems into smaller sections.

• Breaking down problems into smaller parts can make complicated challenges more manageable. This enables other computational thinking elements to be applied more effectively to complex challenges. The solutions to the more minor problems are then combined to solve the original, larger problem.

• Real-world Examples: For instance, when you clean your room, you may put together a to-do list. Identifying the individual tasks (making your bed, hanging up your clothes, etc.) allows you to see the smaller steps before you start cleaning.

2. Pattern Recognition

Recognizing if there is a pattern and determining the relations or sequence etc.

• Examining the problem for patterns, or similarities to previously solved problems, can simplify the solution. Pattern recognition can lead to grouping, organizing, or streamlining problems for more efficient outcomes. Conversely, a lack of patterns is also useful because it means there is no more simplification to be done.

• Real-world Examples: You have likely used pattern recognition in games like UNO, and checkers. Sports like football and basketball also use pattern recognition to identify the opponent's strategy.

3. Abstraction

Generalization of a problem — focus on the big picture and what is essential.

• Taking a step back from the specific details of a given problem allows for creating a more generic solution. This requires analyzing the problem to remove extra detail & highlight the essential parts. then, begin brainstorming a solution to the problem.

• Real-world Examples: Public transportation maps are examples of abstraction that you may encounter often. The maps show only the important information (the stops, the general direction that you are heading) and leave out the finer details.

4. Algorithm

Step-by-step instructions to solve a problem.

• When solving a problem, it is important to create a plan for the solution. Algorithms are a strategy used to determine the step-by-step instructions on how to solve the problem. Algorithms can be written in plain language, with flowcharts, or pseudocode.

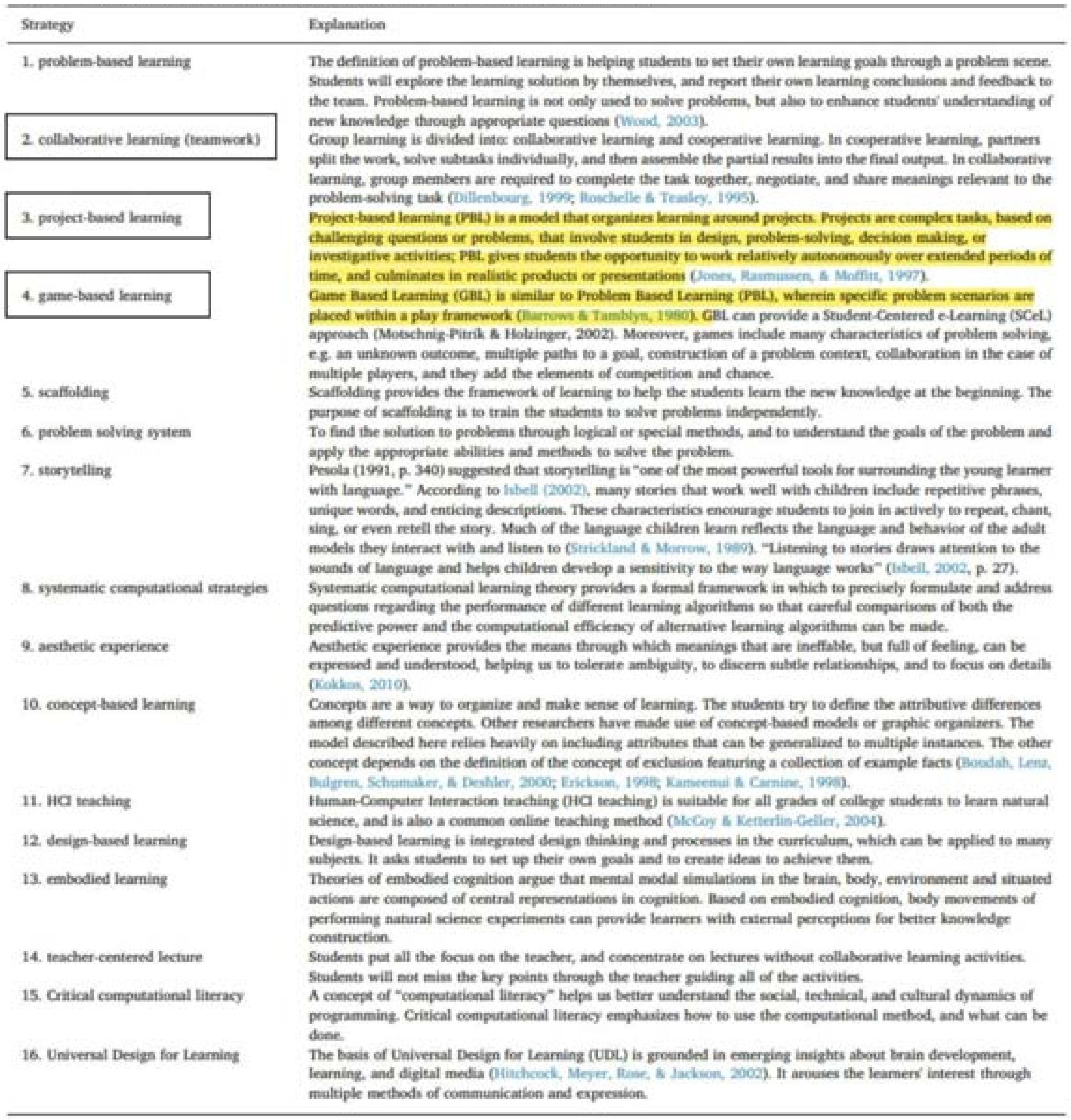
• Real-world Examples: We use algorithms daily, normally in the form of step-by-step instructions. Recipes, instructions for making furniture or building blocks sets, plays in sports, and online map directions are all examples of algorithms.

Ways of Learning CT

*Categories of 16 various learning strategies adopted in the pedagogy of computational thinking skills. Data is taken from the literature review **"How to learn and how to teach computational thinking: Suggestions based on a review of the literature"** [5], a meta-review of the studies published in academic journals from 2006 to 2017.*

*Among these **Project-based learning, collaborative learning, and game-based learning** are the widely used methods to teach CT.*

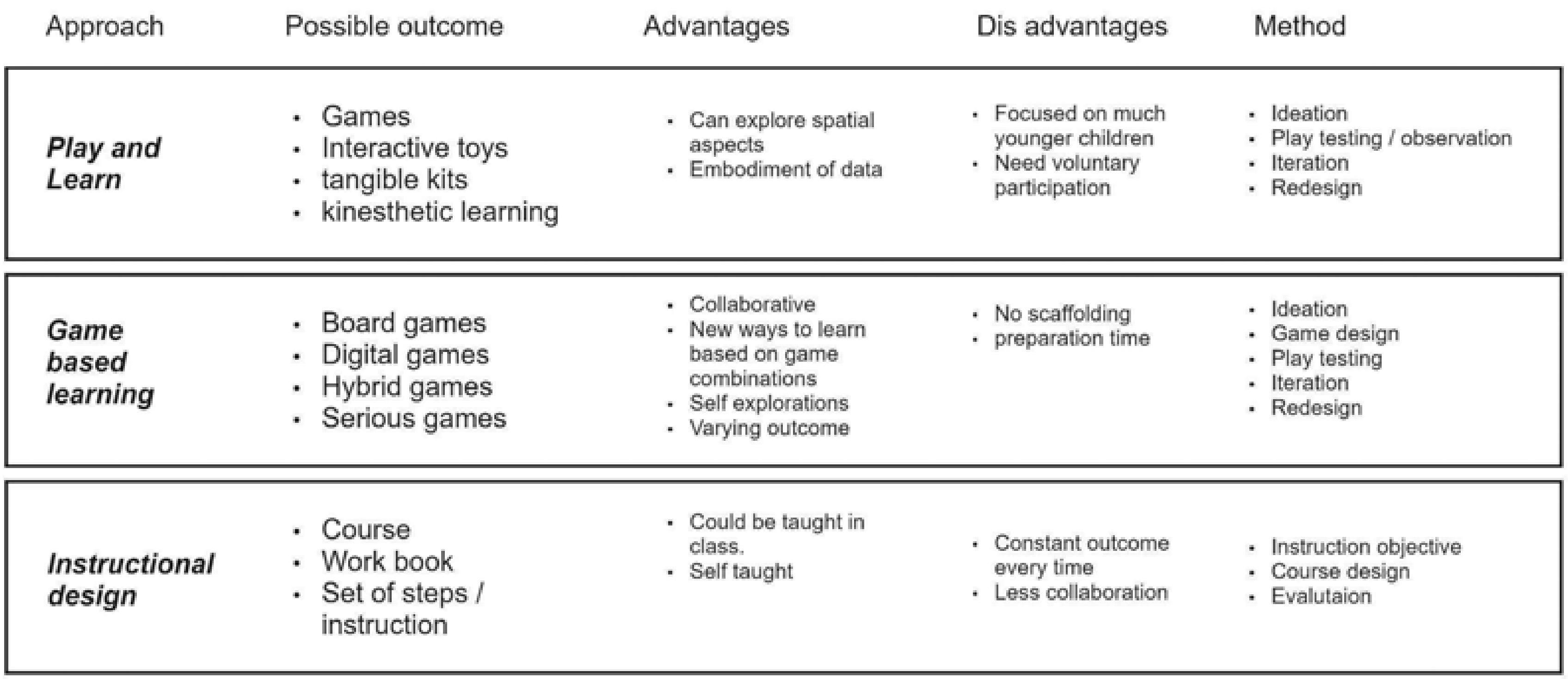
Table 3. Categories of learning.



Strategy	Explanation
1. problem-based learning	The definition of problem-based learning is helping students to set their own learning goals through a problem scene. Students will explore the learning solution by themselves, and report their own learning conclusions and feedback to the team. Problem-based learning is not only used to solve problems, but also to enhance students' understanding of new knowledge through appropriate questions (Wood, 2003).
2. collaborative learning (teamwork)	Group learning is divided into: collaborative learning and cooperative learning. In cooperative learning, partners split the work, solve subtasks individually, and then assemble the partial results into the final output. In collaborative learning, group members are required to complete the task together, negotiate, and share meanings relevant to the problem-solving task (Dillenbourg, 1999; Roschelle & Teasley, 1995).
3. project-based learning	Project-based learning (PBL) is a model that organizes learning around projects. Projects are complex tasks, based on challenging questions or problems, that involve students in design, problem-solving, decision making, or investigative activities; PBL gives students the opportunity to work relatively autonomously over extended periods of time, and culminates in realistic products or presentations (Jones, Rasmussen, & Moffitt, 1997).
4. game-based learning	Game Based Learning (GBL) is similar to Problem Based Learning (PBL), wherein specific problem scenarios are placed within a play framework (Barrows & Tamblyn, 1980). GBL can provide a Student-Centered e-Learning (SCeL) approach (Motschnig-Pitrik & Holzinger, 2002). Moreover, games include many characteristics of problem solving, e.g. an unknown outcome, multiple paths to a goal, construction of a problem context, collaboration in the case of multiple players, and they add the elements of competition and chance.
5. scaffolding	Scaffolding provides the framework of learning to help the students learn the new knowledge at the beginning. The purpose of scaffolding is to train the students to solve problems independently.
6. problem solving system	To find the solution to problems through logical or special methods, and to understand the goals of the problem and apply the appropriate abilities and methods to solve the problem.
7. storytelling	Pesola (1991, p. 340) suggested that storytelling is "one of the most powerful tools for surrounding the young learner with language." According to Isbell (2002), many stories that work well with children include repetitive phrases, unique words, and enticing descriptions. These characteristics encourage students to join in actively to repeat, chant, sing, or even retell the story. Much of the language children learn reflects the language and behavior of the adult models they interact with and listen to (Strickland & Morrow, 1989). "Listening to stories draws attention to the sounds of language and helps children develop a sensitivity to the way language works" (Isbell, 2002, p. 27).
8. systematic computational strategies	Systematic computational learning theory provides a formal framework in which to precisely formulate and address questions regarding the performance of different learning algorithms so that careful comparisons of both the predictive power and the computational efficiency of alternative learning algorithms can be made.
9. aesthetic experience	Aesthetic experience provides the means through which meanings that are ineffable, but full of feeling, can be expressed and understood, helping us to tolerate ambiguity, to discern subtle relationships, and to focus on details (Kokkos, 2010).
10. concept-based learning	Concepts are a way to organize and make sense of learning. The students try to define the attributive differences among different concepts. Other researchers have made use of concept-based models or graphic organizers. The model described here relies heavily on including attributes that can be generalized to multiple instances. The other concept depends on the definition of the concept of exclusion featuring a collection of example facts (Boudah, Lenz, Bulgren, Schumaker, & Deshler, 2000; Erickson, 1998; Kameenui & Carnine, 1998).
11. HCI teaching	Human-Computer Interaction teaching (HCI teaching) is suitable for all grades of college students to learn natural science, and is also a common online teaching method (McCoy & Ketterlin-Geller, 2004).
12. design-based learning	Design-based learning is integrated design thinking and processes in the curriculum, which can be applied to many subjects. It asks students to set up their own goals and to create ideas to achieve them.
13. embodied learning	Theories of embodied cognition argue that mental modal simulations in the brain, body, environment and situated actions are composed of central representations in cognition. Based on embodied cognition, body movements of performing natural science experiments can provide learners with external perceptions for better knowledge construction.
14. teacher-centered lecture	Students put all the focus on the teacher, and concentrate on lectures without collaborative learning activities. Students will not miss the key points through the teacher guiding all of the activities.
15. Critical computational literacy	A concept of "computational literacy" helps us better understand the social, technical, and cultural dynamics of programming. Critical computational literacy emphasizes how to use the computational method, and what can be done.
16. Universal Design for Learning	The basis of Universal Design for Learning (UDL) is grounded in emerging insights about brain development, learning, and digital media (Hitchcock, Meyer, Rose, & Jackson, 2002). It arouses the learners' interest through multiple methods of communication and expression.

Directions to explore

We choose game-based learning/ play and learn approach over the Instructions design approach because games or game-based applications are an increasingly important approach in cognitive training, learning, and educational interventions. It is because of their ability to keep learners motivated to play and interact with the application or learning environment [7]. Well-designed digital games can facilitate learning because they were designed to contextualise learning based on a set of learning principles. Principles such as immediate feedback, sandbox, customization, and adjustable difficulty to motivate players to work within their competence. Also, the current NEP 2020 coding curriculum from 6th to 8th is wholly focused on game-based learning, aligning with future curriculum. It opens an opportunity to try game-based learning approaches to teach computational thinking.



Approach	Possible outcome	Advantages	Dis advantages	Method
Play and Learn	• Games • Interactive toys • tangible kits • kinesthetic learning	• Can explore spatial aspects • Embodiment of data	• Focused on much younger children • Need voluntary participation	• Ideation • Play testing / observation • Iteration • Redesign
Game based learning	• Board games • Digital games • Hybrid games • Serious games	• Collaborative • New ways to learn based on game combinations • Self explorations • Varying outcome	• No scaffolding • preparation time	• Ideation • Game design • Play testing • Iteration • Redesign
Instructional design	• Course • Work book • Set of steps / instruction	• Could be taught in class. • Self taught	• Constant outcome every time • Less collaboration	• Instruction objective • Course design • Evalutaion

Table 4. Directions to explore and approach methods.

3. Literature review

Since we were interested in exploring game-based learning, most of the literature revolves around the play theories and game design approach.

The following literature studied in this section are listed below:

- Piaget’s Theory of Cognitive Development.

- Play theories
  - Caillois’s Attitudes in Play Experience.
  - Csikszentmihalyi’s Flow Theory.
  - Play Pyramid by Kudrowitz and Wallace.

- Learning theory
  - Relationships of computational thinking, pedagogy of programming, and Bloom’s Taxonomy.

- Game design frameworks
  - MDA (mechanics dynamics and aesthetics) framework.
  - DPE framework (Design play and experience).
  - Serious game design framework.

Piaget’s Theory of Cognitive Development

Cognitive Development theory is given by Jean Piaget, who has been one of the most popular and influential experts in the field of developmental psychology and education. According to him, learning occurs in children through the process of adaptation, an active process where children construct knowledge structures through experience and interaction with their environment. It also defines children's cognitive development in various age groups, as mentioned in figure 7.

It has four factors that affect the development process of children [12].

• **Maturation:** exposure to artefacts & technology need to be apt for the age group according to their cognitive development.

• **Experience:** Artifacts in the environment enables them to learn by experiencing different artifacts to form their knowledge naturally.

• **Social Interaction:** Enabling social interactions in the learning environment promotes learning action.

• **Emotions and Motivation:** To learn, learning and developmental activities should be made relevant to children’s lives and interests.

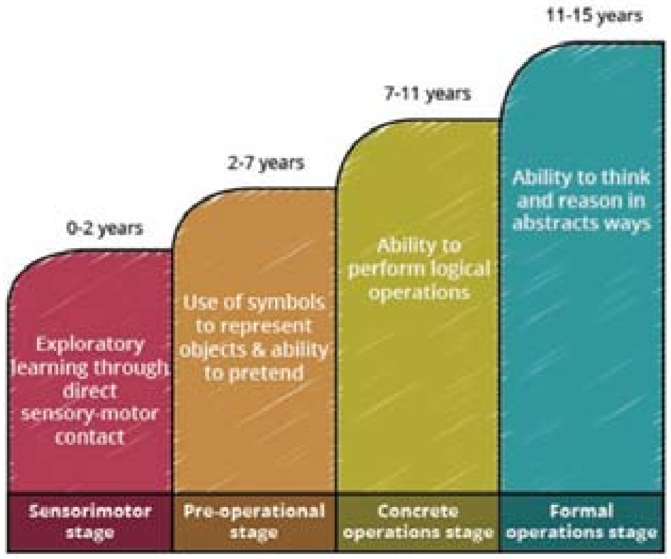

0-2 years
Exploratory learning through direct sensory-motor contact
Sensorimotor stage
2-7 years
Use of symbols to represent objects & ability to pretend
Pre-operational stage
7-11 years
Ability to perform logical operations
Concrete operations stage
11-15 years
Ability to think and reason in abstracts ways
Formal operations stage

Fig 7. Piaget’s Theory of Cognitive Development.

<u>Selection of Age group for the project:</u>
We selected the age group of our project according to Piaget's theory. Piaget’s theory states that the “ability to think and reason in abstract ways” occurs only between 11 to 15 years, which will be necessary to grasp computational thinking concepts.

Theories on Play

Caillois's Attitudes in Play Experience

Caillois identified four broad attitudes in play experiences [10], which lead to four play forms, from completely unstructured, (free play) to structured, (rule-based play) as mentioned in figure 8.

Caillois attitude theory was used to formulate various ideas during brainstorming, it provide opportunities to vary the nature of game and also form new mechanisms in the game play.

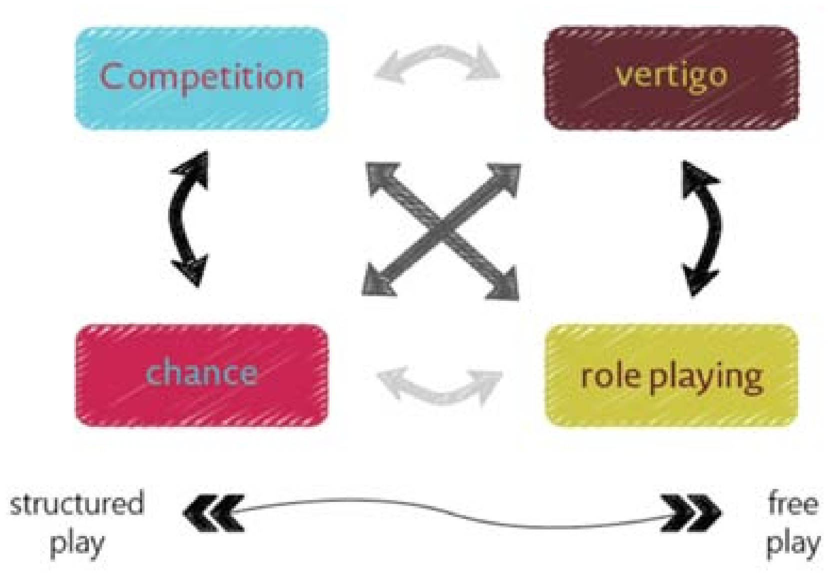


Fig 8. Caillois's Attitudes in Play Experience.

- **Agon (Competition)**: Based on the idea of winning by mastering a single quality/skill against opponents. there is an equal chance of winning. E.g. chess.

- **Alea (Chance):** Based on the idea of winning by favor/luck, rather than skill and experiencing pleasure in the lack of control. E.g. slot machine.

- **Mimicry (Role Playing):** Based on pretending to be another to convince others (audience) such that imaginative reality is maintained. E.g. playing doctor-patient.

- **Ilinx (Vertigo):** Based on the pleasure from altering perception and shocking self through bodily movement. E.g. roller-coaster ride.

<u>How we used it in the project:</u>

The game we presented had part of structured play which contains Agon and Alea style in resource collection mode. An unstructured play which contains aspects of Mimicry during construction mode in the game.

Csikszentmihalyi's Flow Theory

Mihaly Csikszentmihalyi flow theory [11] talks about flow or optimal experience representing a mental state while performing an activity when a person is fully immersed. Flow theory is about adjusting the gameplay states in a graph between challenge and skill to create immersion as shown in figure 9. It is Mostly used in games to create immersion.

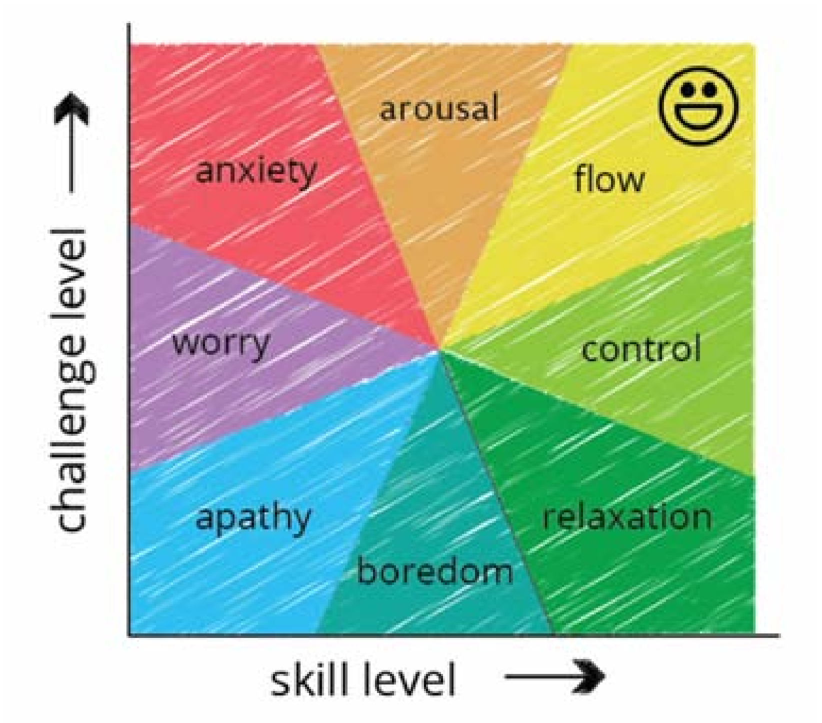


Fig 9. Graph showing skill level vs challenge level in Flow Theory.

According to* Csikszentmihalyi *State of flow can be achieved through:

- Clear goals and immediate feedback when required
- Equilibrium between the level of challenge and personal skill
- Merging of action and awareness
- Focussed concentration
- Sense of potential control
- Loss of self-consciousness
- Time distortion

How we used it in the project:

Flow theory can be used in the project in evaluating an existing play and learning activity and adjusting the level of challenge with respect to the skill or ability of children. It can be used to regulate the difficulty of the content and analyse the ease of game play.

Play Pyramid by Kudrowitz and Wallace

The play pyramid is a three-dimensional map that allows designers to classify a toy concept by placing it in a space between four independent axes representing four types of play, as shown in figure 10. The dimensions are sensory, fantasy, construction and challenge-based play, and it will apply to all age groups [3].

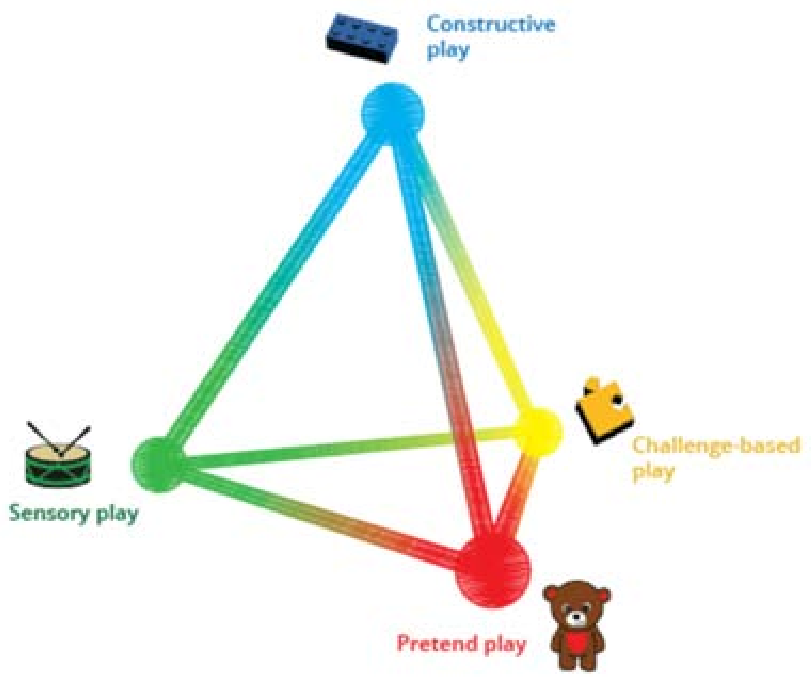


Fig 10. Play Pyramid by Kudrowitz and Wallace.

Play pyramid is a resourceful tool for toy and play designers as they generate different configurations by moving around on the pyramid. A designed concept can also be explored for modification by moving it. For example, lego blocks have been placed in constructive play vertices.

<u>How we used it in the project:</u>

The game designed in the project has part of the constructive play and pretend play, along with challenge based play. This pyramid can be used to analyze how varying one type of play can affect other types, based on which levels of construction and pretend play can be modified in the game.

Theories on Learning

Relationships of CT, pedagogy of programming, & Bloom's Taxonomy

This research paper sums up the relationship between relationships between Bloom's Taxonomy Cognitive Domain, computational thinking skills, & the teaching of programming [4], as shown in figure 11.

The research found that computational thinking skills were perceived to be the most difficult to master.
The following order of perceived difficulty is listed below, with one being the easiest computational skill to master and six being the most difficult.

1. Evaluation
2. Algorithm design
3. Generalisation
4. Abstraction of functionality
5. Abstraction of data
6. Decomposition

<u>*How we used it in the project:*</u>
This relationship can be used to analyze the level in which concept of computational thinking exists and compare the level of learning to Bloom's taxonomy of learning. It will help compare the evaluations of computational thinking.

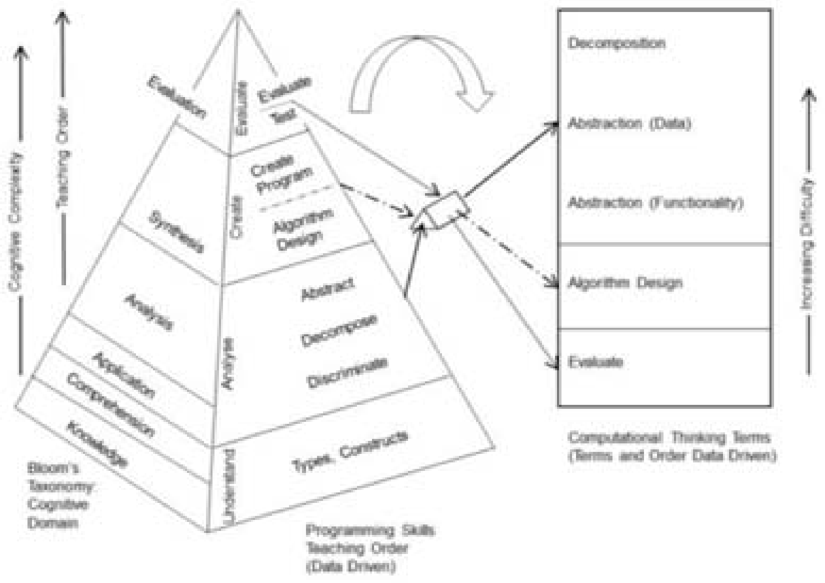


Fig 11. Combined relationship diagram between Bloom's Taxonomy Cognitive Domain, computational thinking skills, and the teaching of programming.

Frameworks of game design

In order to build the game, which generally deals with framing game rules, game mechanics, dynamics, experiences, and playtesting. We went through various game design frameworks, which will help build the game, they were listed below:

- MDA (mechanics dynamics and aesthetics) framework.
- Serious game design framework.
- DPE framework (Design play and experience).

1. MDA (mechanics dynamics and aesthetics) framework

MDA(mechanics dynamics and aesthetics) framework [15] is a formal approach to understanding games that attempts to bridge game design and development, criticism, and technical game research.

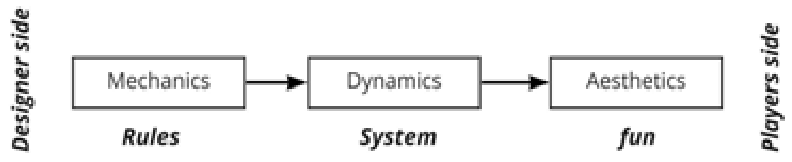


Fig 12. MDA (mechanics dynamics and aesthetics) framework.

Mechanics describes the particular components of the game at the level of data representation and algorithms.

Dynamics describes the run-time behaviour of the mechanics acting on player inputs and outputs over time.

Aesthetics describes the desirable emotional responses evoked in the player, when they interact with the game system.

2. Serious game design framework

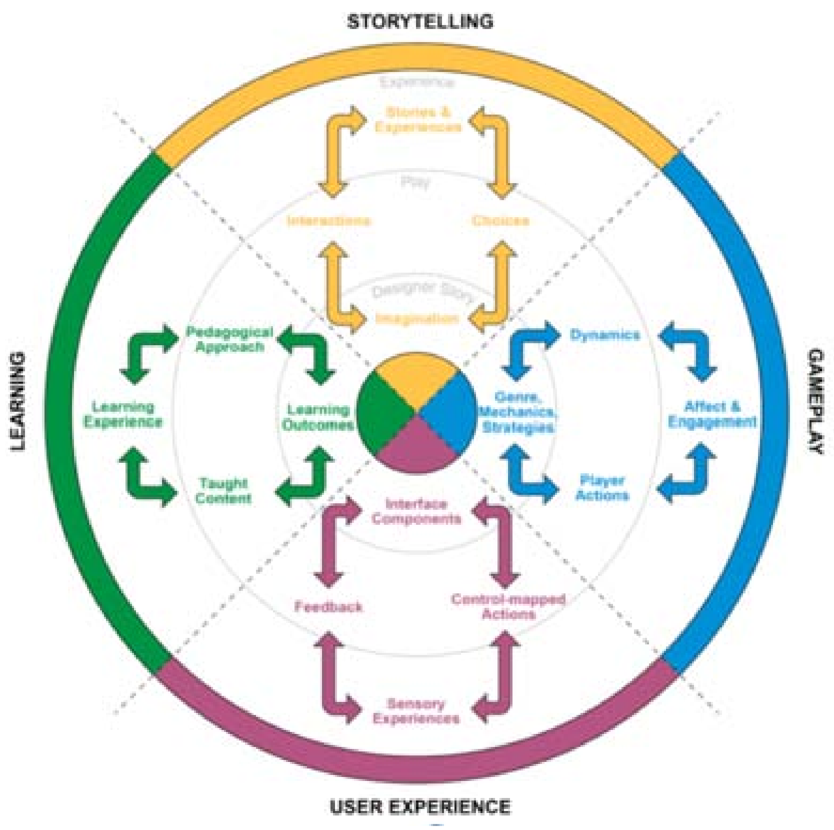


Fig 13. The Art of Serious Game Design by Digital Education Strategies, Ryerson University.

The Serious Game Design conceptual framework, anchored in the Design, Play, Experience Framework [14], is depicted as a circle divided into four equal quadrants, each representing a different but equally important game element: Learning, Storytelling, Gameplay and User Experience. The components within these game elements are connected with double-ended arrows, representing iteration and the interconnectedness between the framework's layers, as shown in figure 13.

It also presents a list of flashcards that facilitates the design process by asking a set of questions for each quadrant of gameplay, learning Storytelling, and User experience.

3. DPE framework (Design play and experience)

The design, play, and experience framework [16], provide a formal approach to designing the learning, storytelling, gameplay, user experience, and technology components of a serious game. The framework provides a common language to discuss serious game design, a methodology to analyze a design, and a process to design a serious learning game.

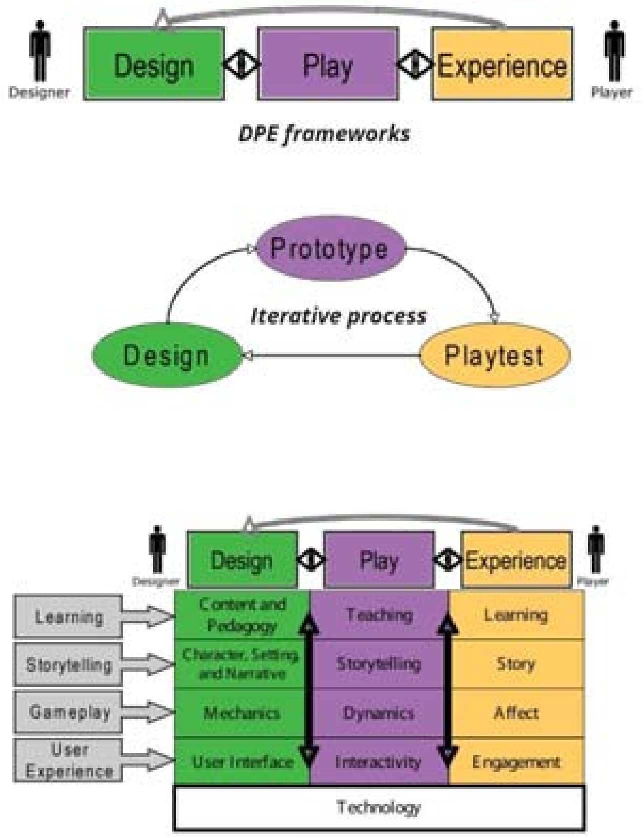


Fig 14. DPE framework (Design play and experience)

4. Secondary Research

Market study

Analyzing products available in the market which focus on game-based learning to teach computational thinking. Most of them focus on coding, not explicitly on computational thinking, so we started analyzing related products where were listed below:

- Tangible products
  - Taco coding by play shift
  - Tangiplay
  - Google Project Bloks

- Visual coding language /interface
  - SCratch
  - MIT App inventor
  - Agent Sheets / Agent cubes

- Games
  - Lightbot
  - CodeSpark
  - Algorithm city
  - Tomorrow corporation’s Human resource machine
  - The case study of Crabs & Turtles
  - RaBit EscApe

Tangible product

Taco coding by play shifu

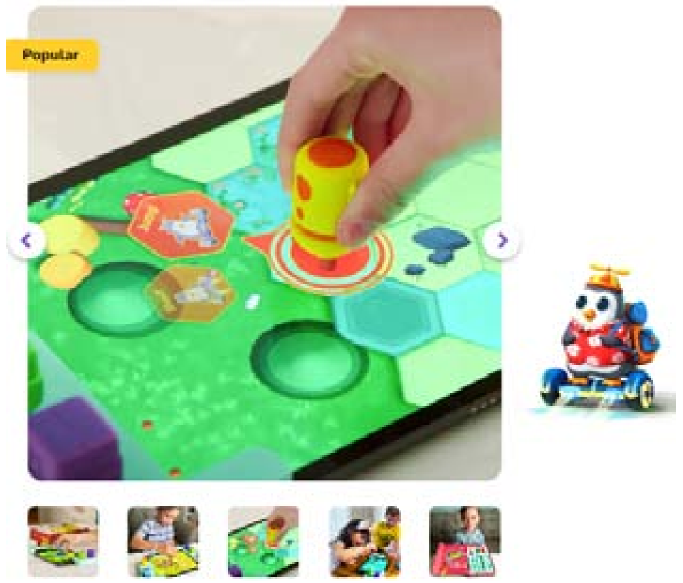


Fig 15. Taco coding App and tangible elements.

Taco is a hybrid product that uses tangible toys and a digital iPad display to provide a combined gameplay experience. It claims to teach the coding ability to children by solving small puzzles. The target audience was young students. Most of the application by taco takes inputs through manipulating the orientation and position of physical toys on the Ipad, as shown in figure 15.

Tangiplay

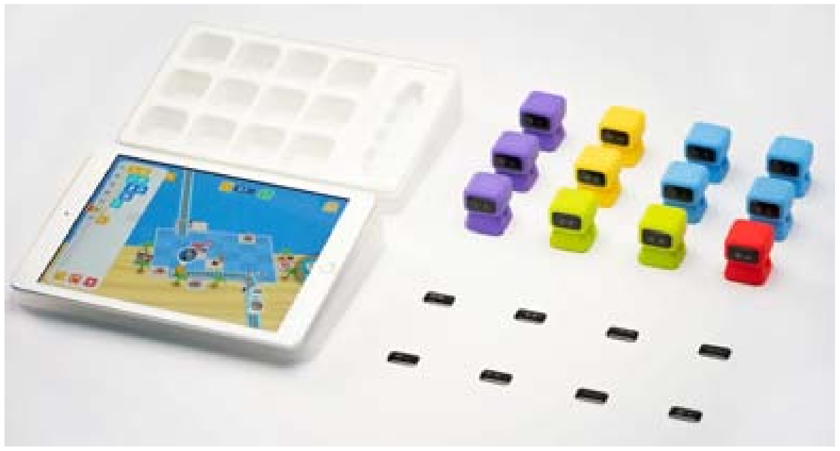


Fig 16. Tangiplay app, along with mini tangible elements.

Tangible coding toy for children of age group 4-12. Similar to Taco, this uses Ipad and customised colour-coded toys using which children instruct the “track building game” in Ipad as shown in figure 16. Building blocks of coding languages like if conditions, loops and instructions were embodied in physical form, which takes input into the game.

Google Project Bloks

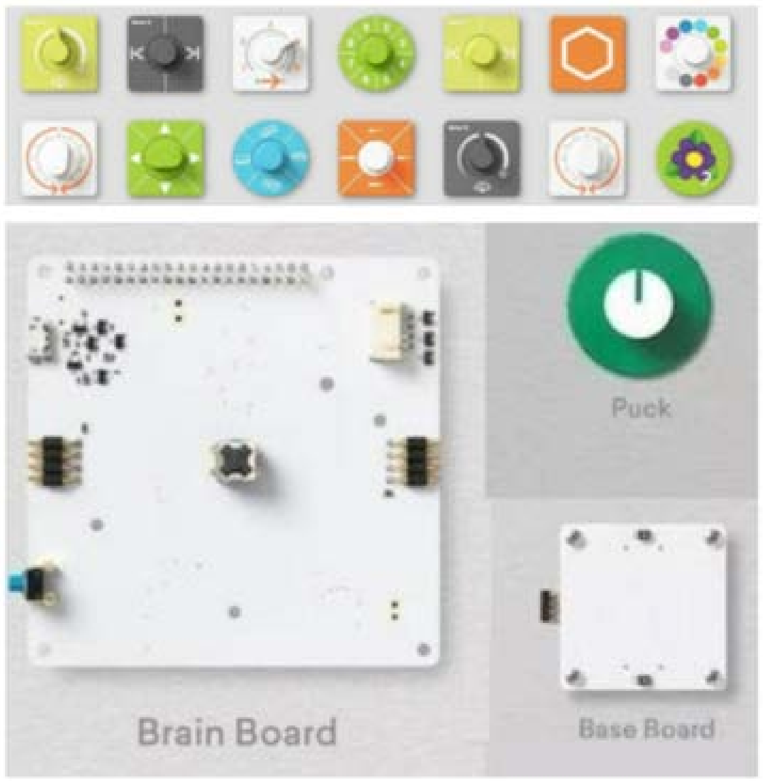


Fig 17. Google project bloks, with all its tangible components.

The project is a collaboration between Google Creative Lab, design consulting firm IDEO and Paulo Bilkstein, Assistant Professor of Education at Stanford University. It consists of pre-programmed blocks. It is a modular design as shown in figure 17. Block is intended to make coding a fun activity for young children by placing it in the context of collaborative play and introducing interactivity with the real world, for example, switching light bulbs on and off.

Visual coding language / interface

SCratch

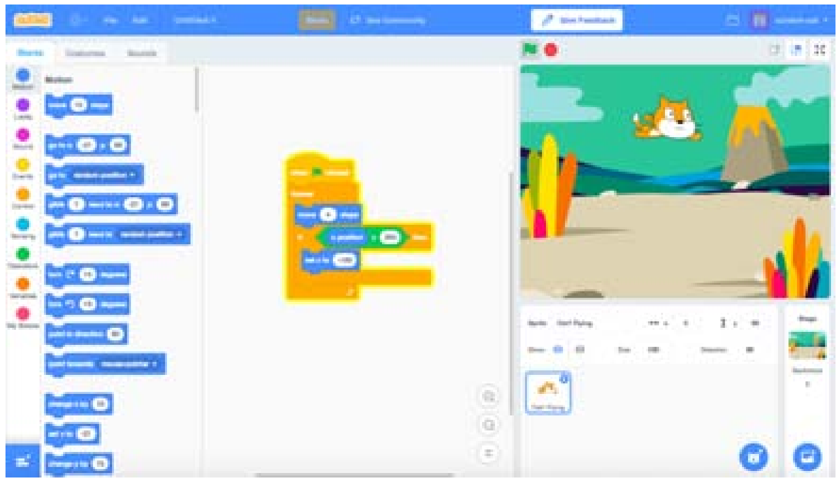


Fig 18. Interface of Scratch programming language.

Scratch is a high-level block-based visual programming language and website targeted primarily at children 8–16 as an educational tool for programming. It is developed by the Massachusetts Institute of Technology (MIT). It has huge potential, a vast open source library and scratch community. It uses blocks based coding language inspired from lego as shown in figure 18. Most of the coding education for early childhood has been taught using scratch in abroad.

MIT App inventor

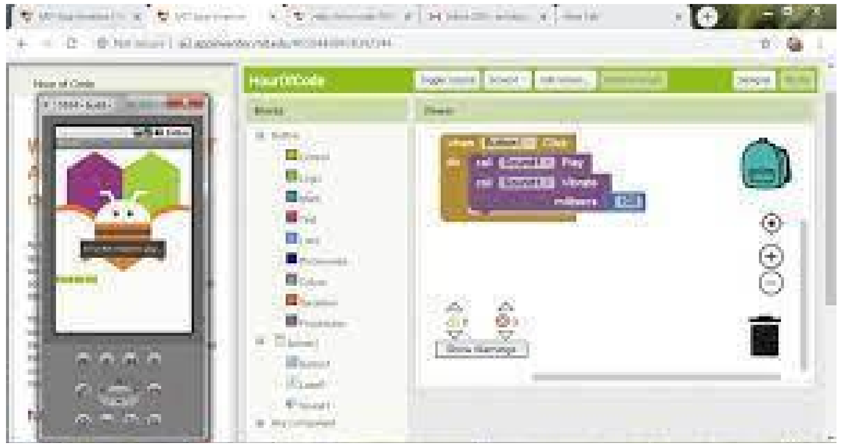


Fig 19. Interface of MIT App inventor.

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software(apps) for two operating systems (OS), android and windows. Itt follows the scratch style of visual coding language, as shown in figure 19.

Agent Sheets / Agent cubes

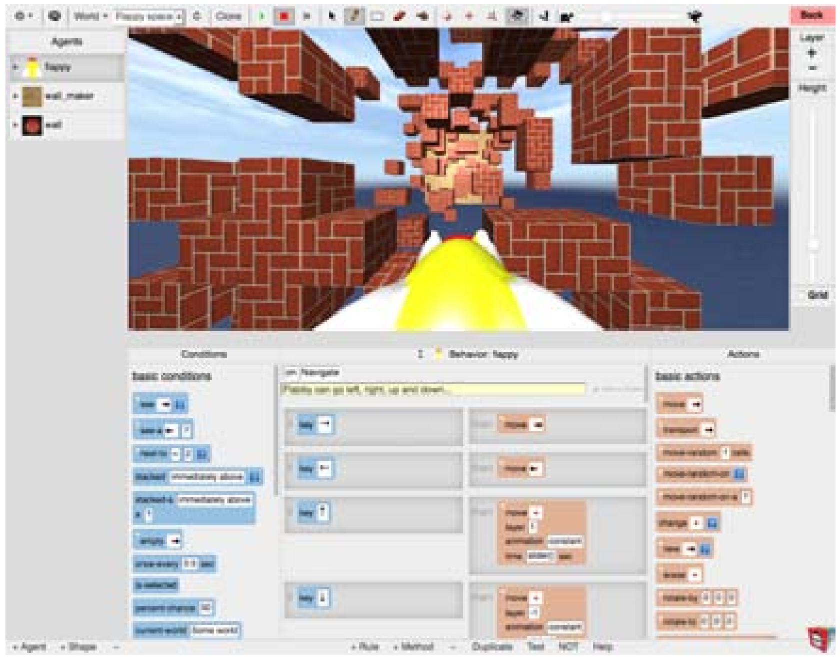


Fig 20. Interface of Agentsheets.

Agent sheets is a powerful Visual Programming Language. Using which kids from 3rd grade on up acquire programming skills for if-then statements, cursor control, movement, loops, collision, diffusion, and many more. Students make games and learn to code, as shown in figure 20.

Games

Lightbot

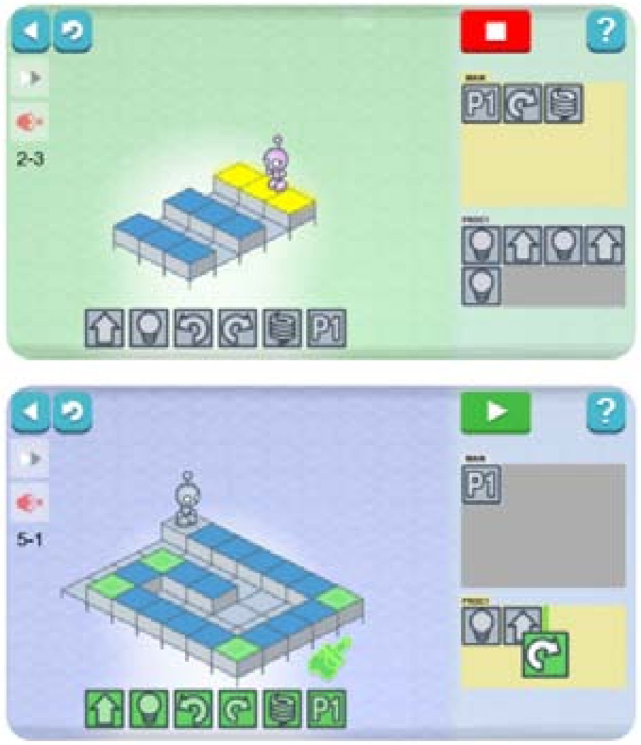


Fig 21. Game play screens of the game Lightbot.

Light bot is a mobile application, in which the player has to drag and drop a set of instructions in the execution panel and then hit play. Game executes it in order, which moves the character accordingly. Objective in the game is to catch the light bulb. This game has a gradual increase in difficulty as the level goes up, it introduces new challenges like conditions and loops as shown in figure 21.

CodeSpark

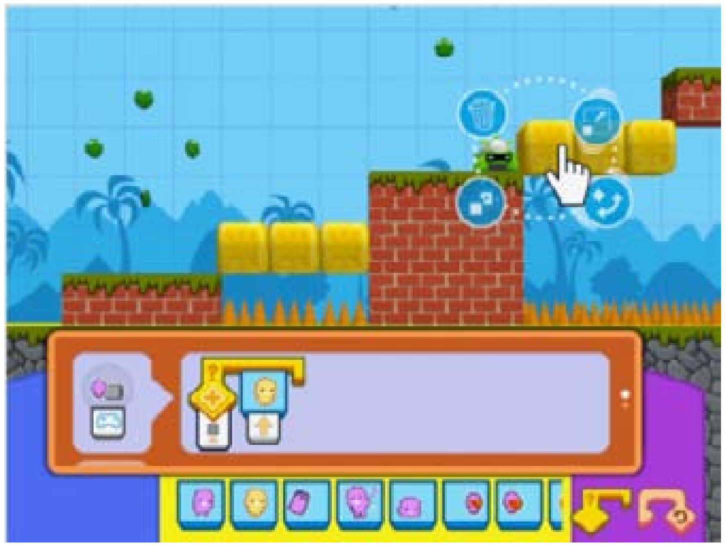


Fig 22. Game play screen of games in CodeSpark.

CodeSpark is a game based learning platform, where children can input their blocks of command in the given space just like Light bot as shown in figure 22. On execution the player moves according to the instructions. It covers almost all aspects of algorithm in computational thinking.

Algorithm city

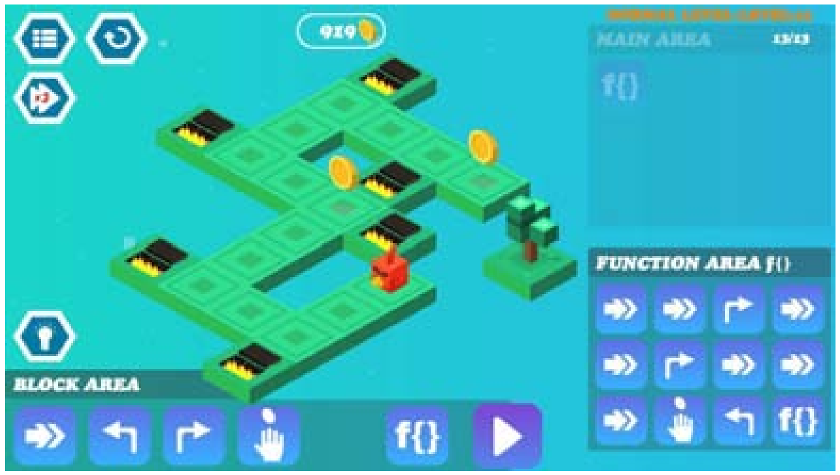


Fig 23. Game play screen of game Algorithm city.

Similar to the first two games. Algorithm city also uses the same game mechanics. Drag and drop instruction model. Only change is the UI and few visual elements.

Human resource machine

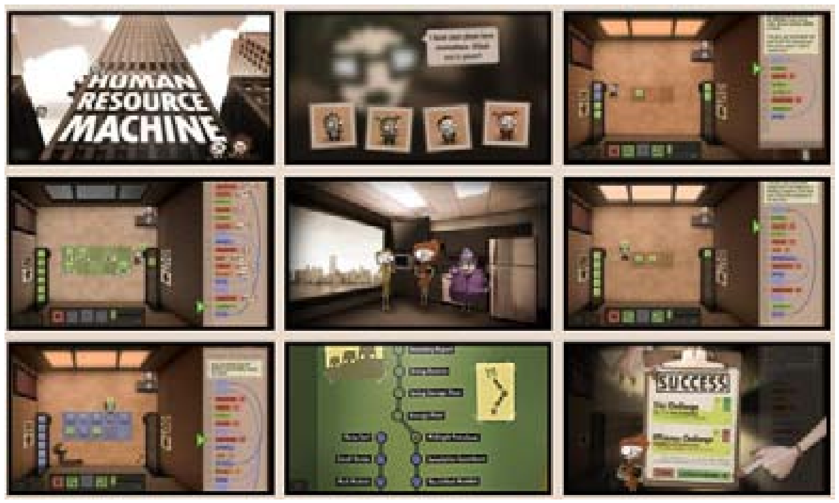


Fig 24. Game play screens of game Human resource machine.

Tomorrow's corporation's Human resource machine is an interesting game which has a lot of features and a good story to keeps player engaged. The player can use scratch like programming language and interact with variables directly. Players can automate works job using coding. This game is engaging and fun to play since it has a good story and visual elements.

The case study of Crabs & Turtles

Training Computational Thinking through board games:

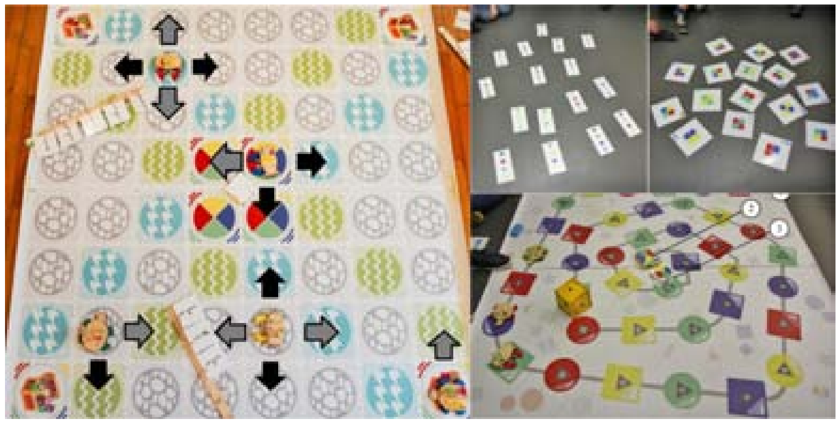


Fig 25. Game play of Crabs & Turtles

It is a paper published in an international journal of serious games [18]. Crabs & Turtles: is a three separate board games, namely the treasure hunt, the race, the Pattern as shown in figure 25, that teaches various concepts separately. It aimed to introduce basic coding concepts and computational thinking processes to 8 to 9-year-old primary school children. The evaluation was performed with adults to validate the design. The treasure hunt game is about restricted movements of carb and turtle where players use a sheet of paper to instruct and move them. 2nd game The Race is a dice and roll game where players uncover small puzzles in each step. 3rd The pattern, which has a list of cards, players have to match the pattern.

RaBit EscApe:

A Board Game for Computational Thinking

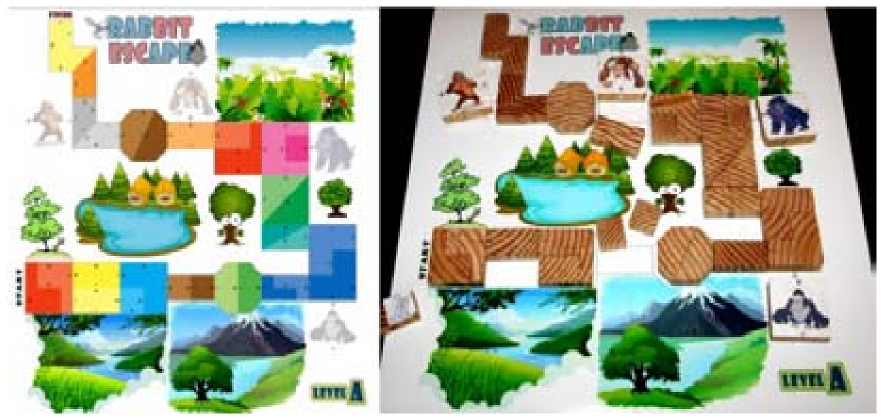


Fig 26. Game play of RaBit EscApe.

It is a paper published in a conference on Interaction design and children [19]. A board game for ages 6 to 10 kids is to orient tangible, magnetised manipulatives to complete or create paths, as shown in figure 26. The game claims to foster children's problem-solving capacity during collaborative gameplay and teach the basics of CT.

Market study insights

- Most of the games either vary input method/output to create an interesting experience.
- Most of the products in the market focus on teaching concepts of coding like conditional statements, loops, variables, data, etc. But not as a whole.
- Some board games might require a game master to assist.
- Most of the digital game in market follows drag and drop mechanism or stack instruction and play mechanism.
- Most of the games teach CT through some form of coding concepts.
- Games which teach CT with real life examples are very rare, so we can use this opportunity.
- There are no full-fledged explicit games to teach computational thinking. In most of the existing games, its implicit.

Informal interview with teachers

Since the students themselves were not aware of computational thinking, getting insight from them was challenging as well as the teachers since they are not practising computational thinking in current cirriculum. So few informal interviews were done with CBSE teachers to understand how computer science was taught in school.

This is not primary research. Users will be involved in the project later during the playtesting and iteration stage.

- Most schools start teaching python directly after the fundamentals and history of computer science.
- Required algorithms were taught only when addressing specific questions in the book.
- Only python IDE was used to teach coding. No additional tools are used.
- The growth of difficulty in programming increases exponentially. They spend time on small commands initially, and later, parts of the subject get less attention.
- Most of the programming problems in the current curriculum are mathematical-oriented.
- There was not enough project-based learning in the current curriculum.

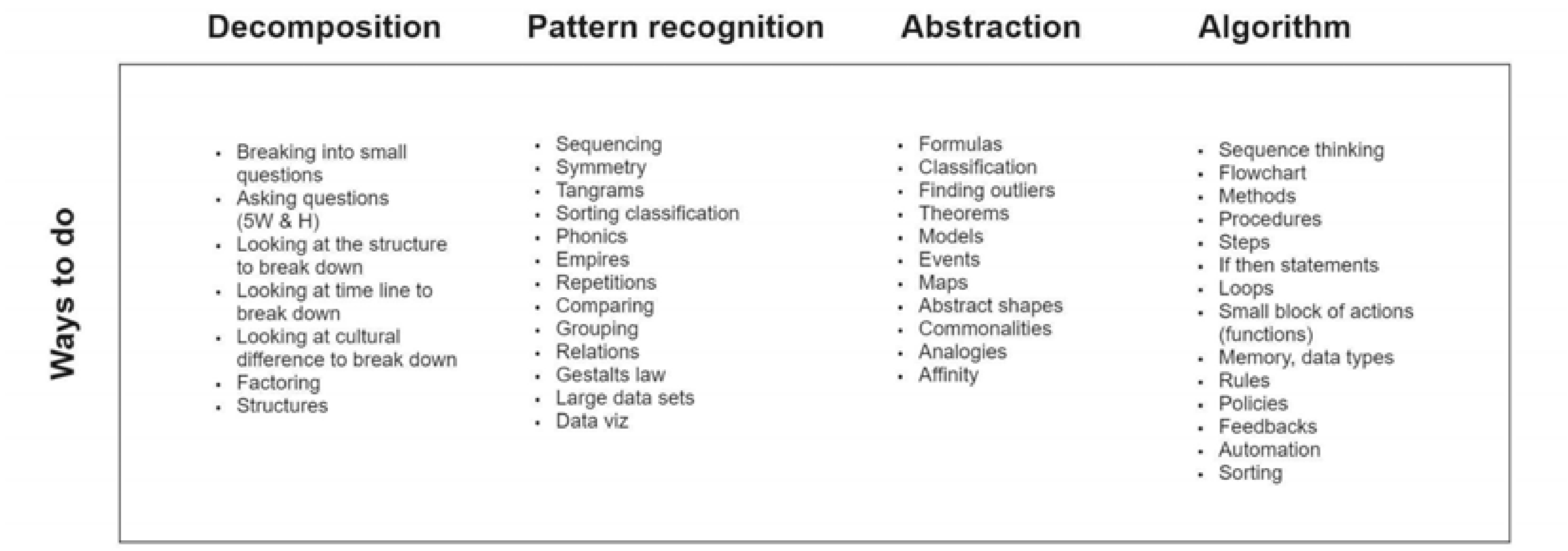
Upon contacting Prof. Sridhar Iyer, from Education technology in IITB. We got a PhD. Student Spruha Satavlekar, who works on computational thinking as a subject matter expert to assist us in this project.

5. Defining

This section presents how we briefed the requirements to create the game. First, we started analyzing the content and how it could be viewed, taught, and various skills needed for achieving it and then figuring out the properties required in the game to teach computational thinking. Later we talked about the learning objectives achieved through the games.

Various ways to do CT

We listed down all the possible ways in which steps of computational thinking (decomposition, pattern recognition, Abstraction, and Algorithms) will be achieved. We decomposed the steps into more minor elements to analyze if we could map any context to teach these four steps, as shown in figure 27.

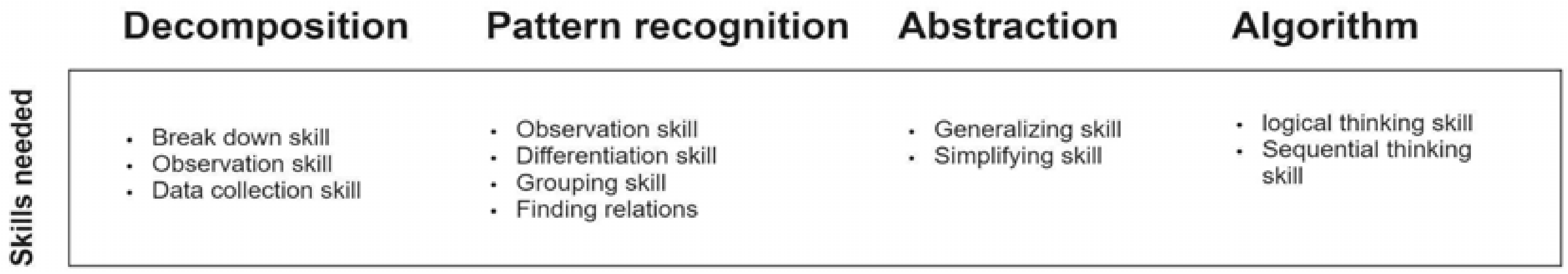


	Decomposition	Pattern recognition	Abstraction	Algorithm
Ways to do	• Breaking into small questions	• Sequencing	• Formulas	• Sequence thinking
	• Asking questions (5W & H)	• Symmetry	• Classification	• Flowchart
	• Looking at the structure to break down	• Tangrams	• Finding outliers	• Methods
	• Looking at time line to break down	• Sorting classification	• Theorems	• Procedures
	• Looking at cultural difference to break down	• Phonics	• Models	• Steps
	• Factoring	• Empires	• Events	• If then statements
	• Structures	• Repetitions	• Maps	• Loops
		• Comparing	• Abstract shapes	• Small block of actions (functions)
		• Grouping	• Commonalities	• Memory, data types
		• Relations	• Analogies	• Rules
		• Gestalts law	• Affinity	• Policies
		• Large data sets		• Feedbacks
		• Data viz		• Automation
				• Sorting

Fig 27. Various ways in which decomposition, pattern recognition, abstractions, and algorithms can be achieved.

Skill needed to perform CT

Defining the skills required to perform (Decomposition, pattern recognition, Abstractions, Algorithms), The game designed should have a list of skills mentioned in figure 28.



	Decomposition	Pattern recognition	Abstraction	Algorithm
Skills needed	• Break down skill • Observation skill • Data collection skill	• Observation skill • Differentiation skill • Grouping skill • Finding relations	• Generalizing skill • Simplifying skill	• logical thinking skill • Sequential thinking skill

Fig 28. Various skills required to perform decomposition, pattern recognition, abstractions, and algorithms.

Requirements to create the game:

Since the concept is to teach itself a way of thinking, i.e., it is a process or procedure that does not have a context on its own, It was challenging to figure out the appropriate content for the game that can accommodate the concepts of computational thinking shown in figure 29.

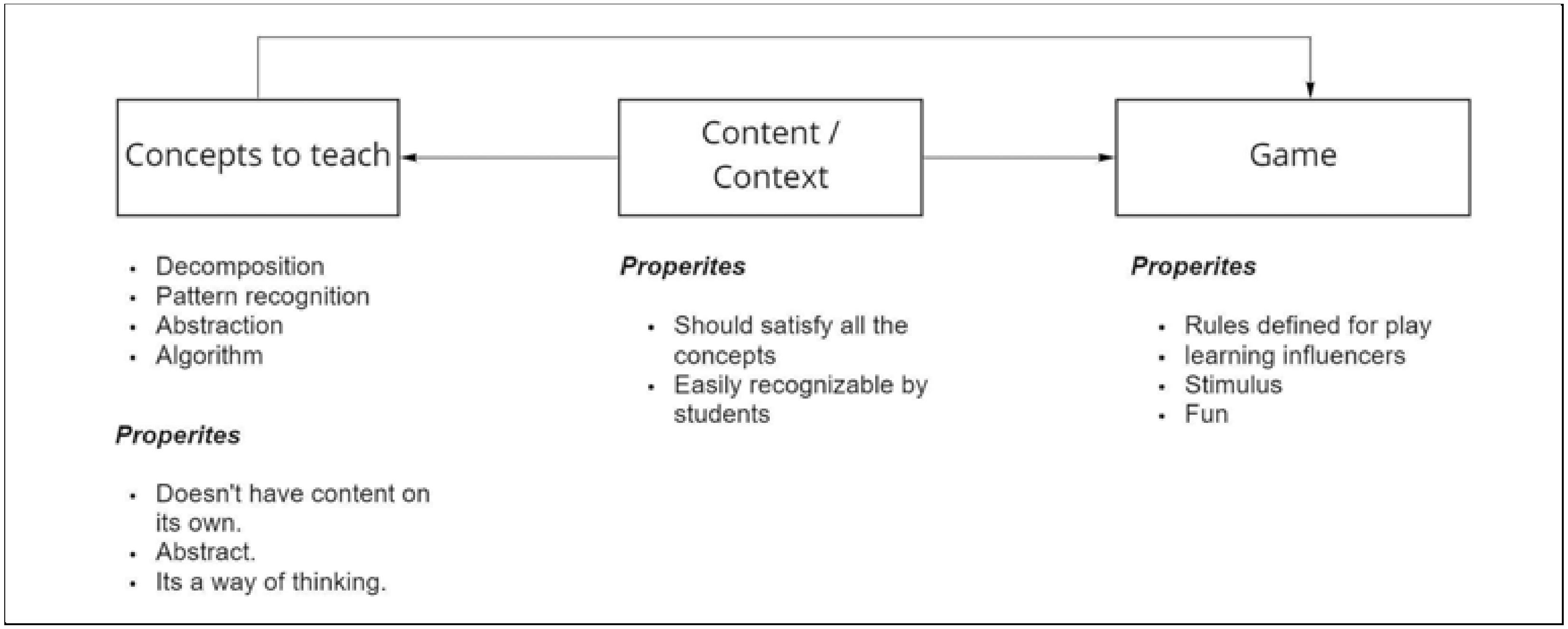
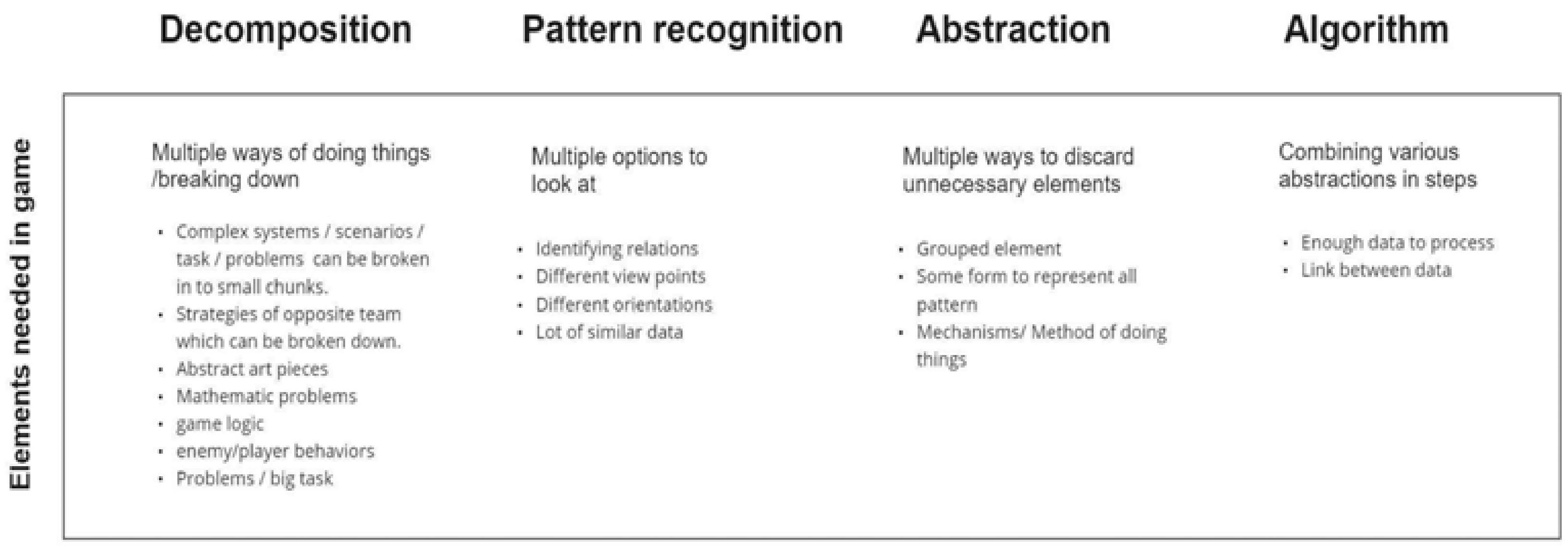


Fig 29. Requirements to create content for the game.

To ideate the game's content, we detailed out the elements to be present in the game, which facilitate the process of CT (Decomposition, pattern recognition, Abstraction, Algorithms), listed in figure 30.



	Decomposition	Pattern recognition	Abstraction	Algorithm
Elements needed in game	Multiple ways of doing things /breaking down • Complex systems / scenarios / task / problems can be broken in to small chunks. • Strategies of opposite team which can be broken down. • Abstract art pieces • Mathematic problems • game logic • enemy/player behaviors • Problems / big task	Multiple options to look at • Identifying relations • Different view points • Different orientations • Lot of similar data	Multiple ways to discard unnecessary elements • Grouped element • Some form to represent all pattern • Mechanisms/ Method of doing things	Combining various abstractions in steps • Enough data to process • Link between data

Fig 30. Various elements must be present in-game to teach decomposition, pattern recognition, abstractions, and algorithms.

Defining learning objectives

Using Gagne/Briggs format, learning objectives were written for each concept of computational thinking.

- Given that the game was played by students (Situation), they will be capable of thinking about how to break down (learning capability) more extensive tasks /problems /scenarios into workable parts (objects) by playing some parts of the game(action).

- Given that the game was played by students (Situation), they will be capable of recognising similarities or commonalities or relationships (learning-capability) between elements (object) by playing some parts of the game (action).

- Given that the game is played by students (Situation), they will be capable of identifying core functional elements (learning capability) from the recognised pattern (object) and applying it to other similar situations (learning capability) by playing some parts of the game (action).

- Given that the game was played by students (Situation), they will be capable of creating steps to solve problems (learning capability) with the available resources (object) by playing some parts of the game (action).

6. Ideations

listed ideas in the upcoming session are not end-to-end ideation. We listed down the ideation of the game ideas. Detailed mechanism and play rules will be defined after the context and learning objective are met.

To solve any problem using Computational thinking, it is necessary to follow the steps of Computational thinking in the order of (Decomposition, pattern recognition, Abstractions, and Algorithms). However, to teach the concepts, the order can be shuffled to teach the concepts, and when applying in-game, players can or cannot follow the sequence.

We started Ideating based on computational thinking concepts individually and combinedly. Some of the ideas are listed below:

Ideation based on topic

1. Pattern recognition using tangrams
2. “Shoot a question”, to lean decomposition
3. “Instruct” to teach Algorithm

Ideation based on context

4. A new Crafting system
5. Computer science game
6. “Object forming” as a game
7. Construction game

Ideation based on topic

1. Pattern recognition using tangrams

Game elements

- Picture cards
- Or tangrams (shapes)
- Or transparent picture cards
- Draw pouch with all elements

How the game works:

- There will be a context card that reveals the common theme for the round.
- In each round, players can take some of the tangrams, or they can play the tangrams on the desk.
- They can stack the tangrams in their hand in turns.
- A team which forms the tangrams based on the context win as shown in figure 31.

Possible Learning outcomes:

- **Visual pattern:** when players arrange various tangrams, they will be able to identify visual patterns of tangrams.
- **Orientations:** Players will get geometry concepts like symmetry, orientations, rotations, etc.

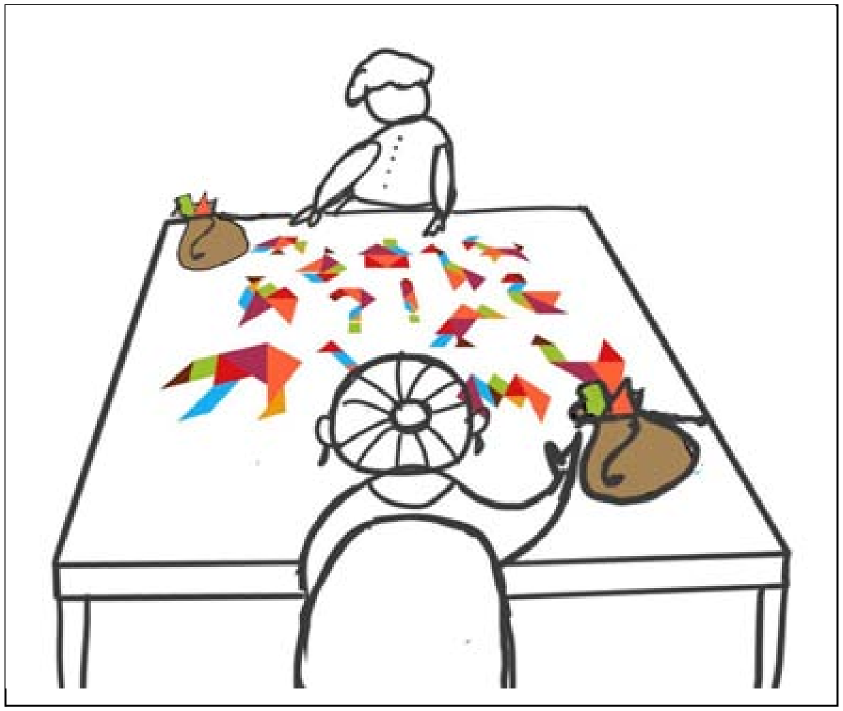


Fig 31. Tangram game.

2. “Shoot a question”, to lean decomposition

Game elements:

- **Context/scenarios cards:** Question or problem (eg: fire accident, in cotton factory)
- **Answer deck:** list of Answer cards for the scenario (eg: like use water, use sand, etc..)

How the game works:

- It is a team game, a team size of 2
- Each team picks a list of Context cards.
- A team asks another team questions based on the context card on each turn.
- Playing team answers with respective cards in their hand.
- Each answer card has different points, so the opposite team has to break down the scenarios and ask questions based on that.
- At the end of each round, points were calculated, and the team that gathered more points wins.

Possible Learning outcomes:

- Question decomposition:
    - Players will be able to think of multiple sub scenarios
    - As the game continues, players will learn how to ask important questions.

3. “Instruct” to teach Algorithm

Game elements:

- **Situation cards:** (e.g., how to brush)

How the game works:

- It is a team game, a team size of 2.
- One person in the team acts like a robot, and another person has to instruct them on the situation opposite team dictates.
- Exact step-by-step instruction has to be given.
- The opposite team will check if they perform anything more than they were instructed.
- If the robot person of the team overrides the instruction, their team losses.
- Each turn, players pick up the situation card and instruction happens within the stipulated time.

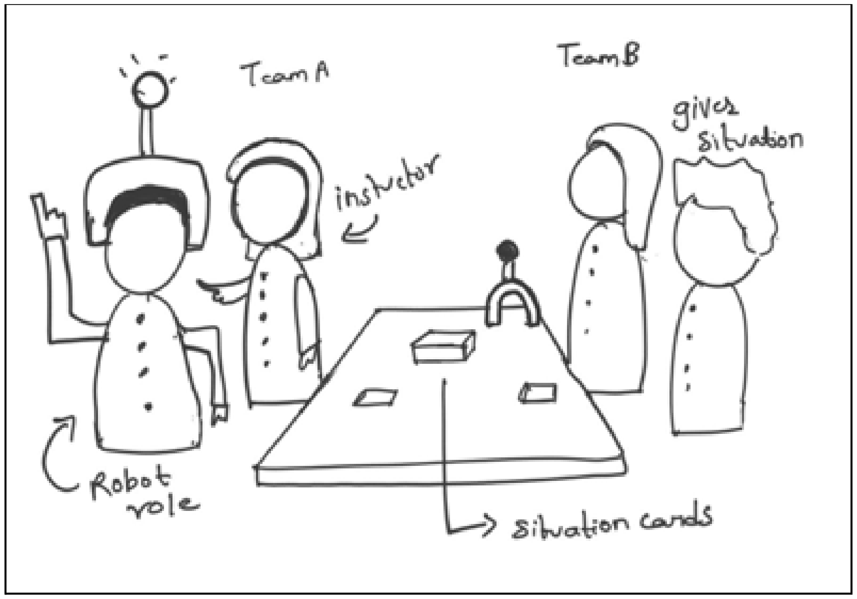


Fig 32. Instruct game idea

Possible Learning outcomes:

- Task decomposition skill:
  - Players will be able to think of how to approach a task in small steps before instructing.
- Sequential thinking skill:
  - Players will be able to, from steps of instruction, to instruct teammates.
- Communication skills:
  - Players will gain communication skills since the game involves many conversations.

Ideation based on context

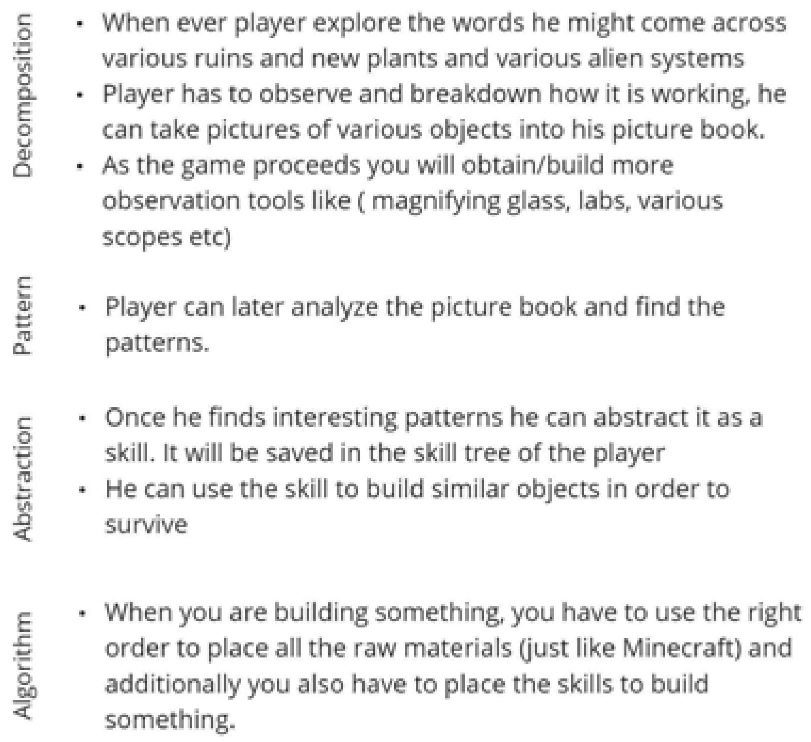
4. A new Crafting system

Game setting:

- **Single-player digital game**
- **World:** Alien planet
  - Various types of ruins
  - Various types of vegetation
- **Type:** Exploratory and survival type

How the game works:

- It is a single-player game similar to Minecraft, but it has an entirely novel crafting system that teaches computational thinking.
- The player will be stranded on an alien planet and trying to understand the world and survive.
- Refer to figure 33, for example.



Decomposition	• When ever player explore the words he might come across various ruins and new plants and various alien systems • Player has to observe and breakdown how it is working, he can take pictures of various objects into his picture book. • As the game proceeds you will obtain/build more observation tools like (magnifying glass, labs, various scopes etc)
Pattern	• Player can later analyze the picture book and find the patterns.
Abstraction	• Once he finds interesting patterns he can abstract it as a skill. It will be saved in the skill tree of the player • He can use the skill to build similar objects in order to survive
Algorithm	• When you are building something, you have to use the right order to place all the raw materials (just like Minecraft) and additionally you also have to place the skills to build something.

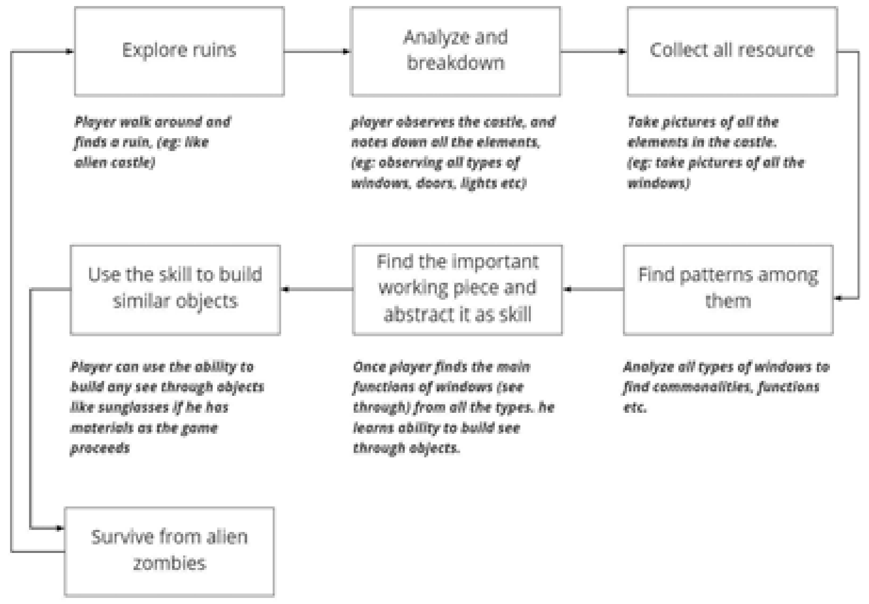

Explore ruins
Analyze and breakdown
Collect all resource
Player walk around and finds a ruin, (eg: like alien castle)
player observes the castle, and notes down all the elements, (eg: observing all types of windows, doors, lights etc)
Take pictures of all the elements in the castle. (eg: take pictures of all the windows)
Use the skill to build similar objects
Find the important working piece and abstract it as skill
Find patterns among them
Player can use the ability to build any see through objects like sunglasses if he has materials as the game proceeds
Once player finds the main functions of windows (see through) from all the types. he learns ability to build see through objects.
Analyze all types of windows to find commonalities, functions etc.
Survive from alien zombies

Fig 33. Gameflow diagram with an example.

Learning outcomes:

- Observation skill:
  - Players have to observe new elements in the game to break them down since they are new and unfamiliar objects.
- Pattern recognition:
  - the player has to find commonalities in the objects collected to form a new craft skill
  - Also, during crafting, materials have to be placed in a particular pattern to form items, just like in the game mine-crafts
- Algorithm
  - Crafting objects in the game requires sequential thinking.

Drawbacks:

- We were not considering this idea because the game's complexity demands computer generation, making it a single-player digital game.
- Learning through collaborative aspects will be lost.
- And not enough time to prototype such complex game mechanisms.

5. Computer science game

I am abstracting the whole coding concept into a card game to teach computational thinking.

Game elements:

- **Data cards:** Contains numbers, variables, and strings (all rewritable)
- **Operator cards:** Mathematical operators
- **Variable / memory cards:** Types of memory blocks
- **Condition cards:** Greater than, lesser than, equal, unequal
- **Abstracted function cards:** rewritable cards to store small program blocks
- **Loop cards:** creates a loop of functions
- **Goal cards:** small programs as task

How the game works:

- Players play in teams of 3 or 2 members
- A player picks goal cards for around (which are small programs, eg: find the max of 3 numbers)
- In each turn, the player has to draw cards and play a few cards on the desk
- The round continuous until they finish the logic
- Each player's turn they can counter the action of another player by placing respective cards, or they can build their logic
- Players can abstract some stack of instructions into abstract cards and use it later.
- Team that tries to create a logic first wins the game.

Learning Outcomes:

- Computer science knowledge:
  - Players will be able to understand the functions of essential coding elements.
- Math:
  - Since most of the programming deals with simple mathematical problems.
- Logical thinking skills:
  - by playing with logic cards and forming small logic in the games
- Strategy breakdown skill:
  - Players will gain the ability to analyse how other players move and recognise their play patterns and how to counter them.
- Problem breakdown:
  - Players have to think about how to attain smaller parts for the given problems (e.g., if the program is to find a greater of 3 numbers, they have to think about what all they need, i.e., three memory to store, comparison cards, etc..)
- Algorithms:
  - The whole game involves performing algorithms in turn by turn order.
- Abstraction Skill:
  - By reusing the same cards they play, like creating small predefined functions.

6. “Object forming” game

Game elements:

- **Context cards:** daily life things
- **data cards:** resources
- **abstract cards:** mechanisms/functions
- **Algorithm cards:** Instructions (like screw it, weld it)
- **arrangement cards:** relationships
- **Each card has its own deck.**

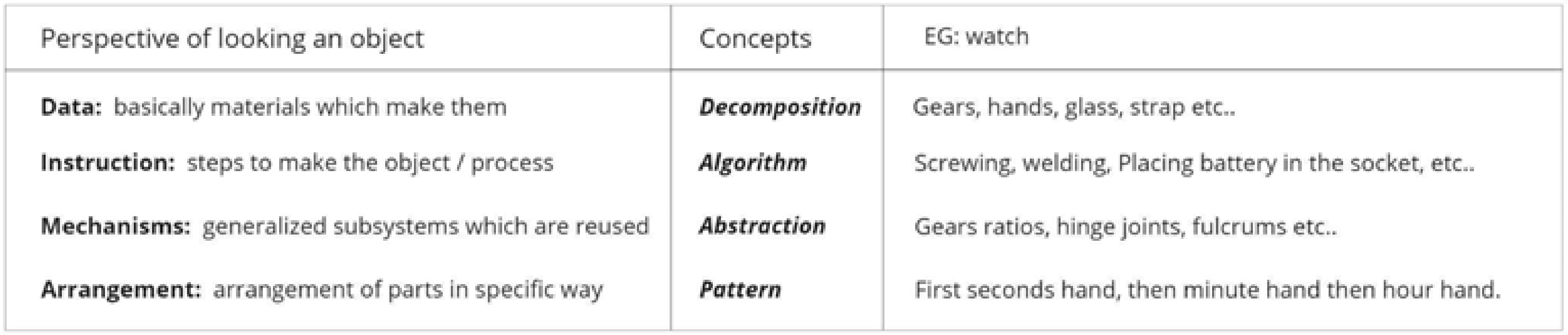
Any objects can be broken down into data, arrangements, instructions, and mechanisms, as shown in Table 7.

How it works:

- It is a team game. Players form 2 people team.
- Each player, in their turn, either picks cards from each deck or trade cards.
- The goal is to complete as many contexts/objects as possible.
- Game mechanics follow the trading of cards
- Whoever finishes the context first wins.

Possible Learning outcomes:

- Resource breakdown:
    - Players break down objects into smaller parts with prior knowledge.
- Strategy breakdown skill
    - Players will gain the ability to analyse how other players trade cards and counter them by restricting required resources.
- Collaboration skill
    - It is a team game, so players have to deal with resources within the team to finish the objects.
- Manufacturing processes:
    - Players will be able to understand how objects were made as they play the game.



Perspective of looking an object	Concepts	EG: watch
Data: basically materials which make them	***Decomposition***	Gears, hands, glass, strap etc..
Instruction: steps to make the object / process	***Algorithm***	Screwing, welding, Placing battery in the socket, etc..
Mechanisms: generalized subsystems which are reused	***Abstraction***	Gears ratios, hinge joints, fulcrums etc..
Arrangement: arrangement of parts in specific way	***Pattern***	First seconds hand, then minute hand then hour hand.

Table 5. “Object forming” game perspective.

7. Construction game

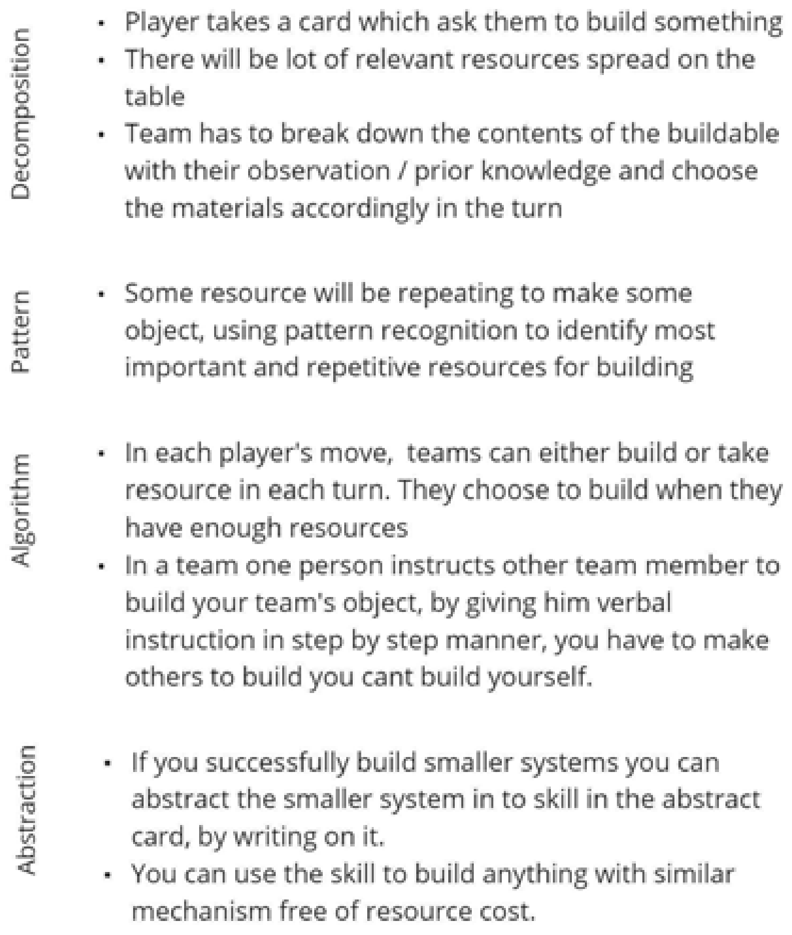
Another form of the previous idea which involves the context of the daily object in a construction game.

Game Setting:

- **Buildable cards:** Daily life things (goal is to build, eg: watch)
- **Resources cards:** All materials required to build ((gears, hands, glass, leather, rubber, battery))
- **Clue cards:** in case people get stuck
- **rewritable cards: acts as abstract cards**, over which people can write anything.

How it works:

- It is a team game (2 per team)
- The team's goal is to build objects, a team that creates the first three objects wins.
- Each turn, players can either.
    - Take a buildable card (max four only can be taken by a player)
    - Take resources
    - Instruct a teammate and make him build
    - abstract some built subsystem into abstract skill



Decomposition	• Player takes a card which ask them to build something • There will be lot of relevant resources spread on the table • Team has to break down the contents of the buildable with their observation / prior knowledge and choose the materials accordingly in the turn
Pattern	• Some resource will be repeating to make some object, using pattern recognition to identify most important and repetitive resources for building
Algorithm	• In each player's move, teams can either build or take resource in each turn. They choose to build when they have enough resources • In a team one person instructs other team member to build your team's object, by giving him verbal instruction in step by step manner, you have to make others to build you cant build yourself.
Abstraction	• If you successfully build smaller systems you can abstract the smaller system in to skill in the abstract card, by writing on it. • You can use the skill to build anything with similar mechanism free of resource cost.

Possible Learning outcomes:

- It can teach all steps of Computational thinking (Decomposition, pattern recognition, algorithm, abstraction)
- Resource breakdown:
    - Players break down objects into smaller parts with prior knowledge.
- Strategy breakdown skill:
    - Players will gain the ability to analyse how other players collect resources and restrict it.
- Collaboration skill:
    - It is a team game, so players have to deal with resources within the team to finish the objects.
- Manufacturing processes:
    - Players will be able to understand how objects are made as they play the game.
- Sequential thinking skill:
    - Players will be able to form steps of instruction to instruct their teammates.
- Communication skill:
    - Players will gain communication skills since the game involves many conversations.

Feedbacks:

- Players can also invent some innovative objects not in the context card during the gameplay.
- How do they visualise constructing objects?
- The situation can be given to analyse instead of objects. Situations and scenarios will have more options for students to think about while constructing objects with prior knowledge .

Idea Selection

Out of all the ideas mentioned above, the **seventh idea, "construction game,"** seems to have the potential for the listed reasons:

- This idea can accommodate all the aspects of Computational Thinking (Decomposition, pattern recognition, Abstractions, Algorithms)
- Also, the flexibility of forming rules around the context of the construction game.
- It is a team game so it has possibilities for collaboration and discussion
- Players can decode other team's strategies, which creates competitive gameplay
- Out of computer science context.
- Outside any curriculum context, it teaches with day-to-day objects that students easily recognise.
- Possibility of including various other technologies / tangible items into the gameplay.
- The same concept can be extended into a game with day-to-day scenarios instead of objects.
- More opportunities for other kinds of learning.
- Also, this will be a cards/board game, which gives a temporal advantage in prototyping and testing compared to digital considering the short period of this project.

Moreover, mixing and matching some of the other above mentioned ideas were also used to create the game.

7. Content design of the game

Content of the construction game was designed considering the day to day life activities of students in high school. Such that they are familiar with the items they are going to build, and they can focus on the mechanism and gameplay without spending time learning about the content.

To understastudents't's interest and day to day activity, we floated survey questions as shown in figure 34. The survey had questions like.

- The game they play.
- How often they play games.
- Board games they played.
- An educational game they played.
- Interesting day-to-day activity.
- Things they carry/use in school .
- Types of punishment they see in schools.

Contents of 'interesting day to day activities' and 'things they use in day to day activities' are considered to make scenario cards in the game, along with the respective resources cards required to solve them.

Content for the game was developed by considering the daily objects, building scenarios around them, and breaking them down to find the resources required and later grouped all the similar scenarios. The list of scenarios, resources, and content has been mentioned in the excel sheet, as shown in figure 35.

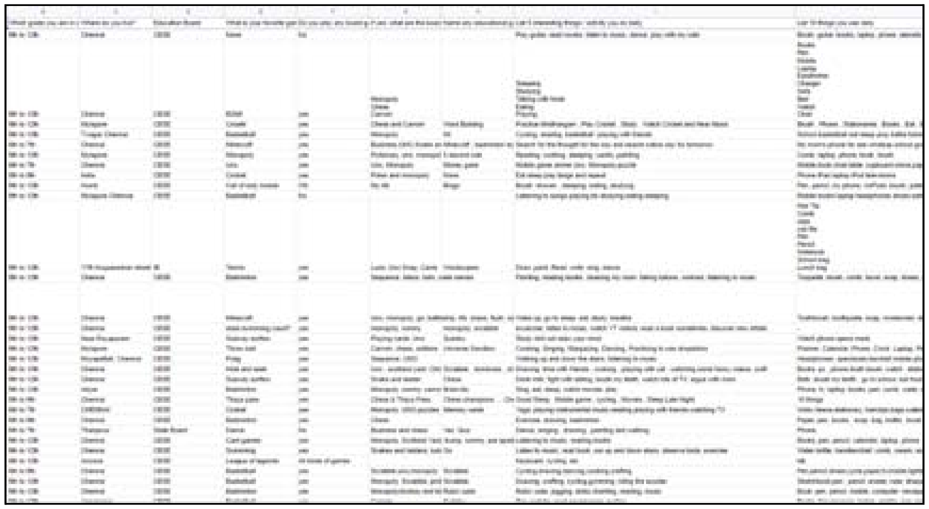
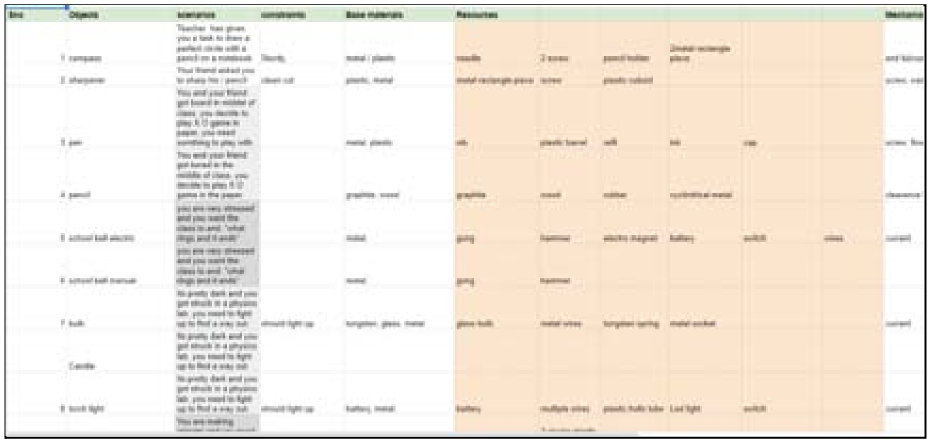


Fig 34. Excel sheet containing the student interest data.



Sno	Objects	scenarios	constraints	Base materials	Resources						Mechanism
	1 compass	Teacher has given you a task to draw a perfect circle with a pencil on a notebook	Sturdy	metal / plastic	needle	2 screw	pencil holder	2metal rectangle piece			[illegible]
	2 sharpener	Your friend asked you to sharp his pencil	clean cut	plastic, metal	metal rectangle piece	screw	plastic cuboid				screw, [illegible]
	3 pen	You and your friend got bored in middle of class, you decide to play X O game in paper, you need somthing to play with		metal, plastic	nib	plastic barrel	refill	ink	cap		screw, [illegible]
	4 pencil	You and your friend got bored in the middle of class, you decide to play X O game in the paper		graphite, wood	graphite	wood	rubber	cylindrical metal			clearance [illegible]
	5 school bell electric	you are very stressed and you want the class to end. "what rings and it ends"		metal	gong	hammer	electric magnet	battery	switch	wires	current
	6 school bell manual	you are very stressed and you want the class to end. "what rings and it ends"		metal	gong	hammer					
	7 bulb	Its pretty dark and you got struck in a physics lab, you need to light up to find a way out	should light up	tungsten, glass, metal	glass bulb	metal wires	tungsten spring	metal socket			current
	Candle	Its pretty dark and you got struck in a physics lab, you need to light up to find a way out									
	8 torch light	Its pretty dark and you got struck in a physics lab, you need to light up to find a way out	should light up	battery, metal	battery	multiple wires	plastic bulb tube	Led light	switch		current
		You are making [illegible]				[illegible]					

Fig 35. Excel sheet containing the content of the game.

8. Game Design

Game instruction link / Gameplay video link

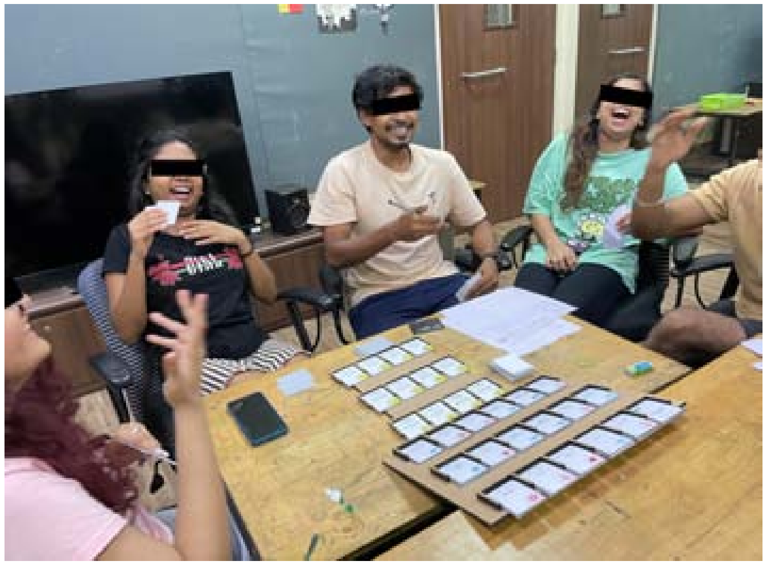


Fig 36. Game play with our batch mates.

Game Design

This is a Construction game where players compete in teams to build objects. The game has two parts. One is **resource collection,** and the next is the **building mode(algorithm mode)**. The following section explains the game in detail. **This game has matured over various rounds of iterations.**

The following are explained in this section:

- Game placement.
- Game elements.
- Game setting.
- Game flow.
- Results of the game.
- Game from the player's prespective.
- Where and when they learn CT in the gameplay.

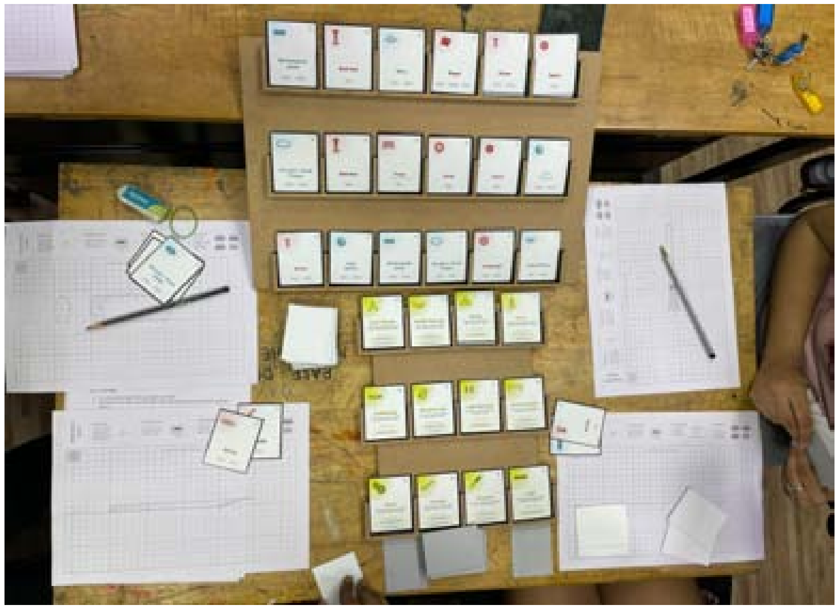


Fig 37. Game play

Game Placement

This game was designed to be **played in a workshop** with **some game master around**, although this game can also be played without a game master. Ideal game play time was around 1hr. The game can be used along with the other computational thinking activities also. Game master or referee was required since the players were school students and some inputs are needed to resolve conflicts now and then or to guide them when they go wrong somewhere.

Fig 38. Game play testing with school students.

Game Elements

This excel sheet contains the list of objects and scenarios considered in the game.

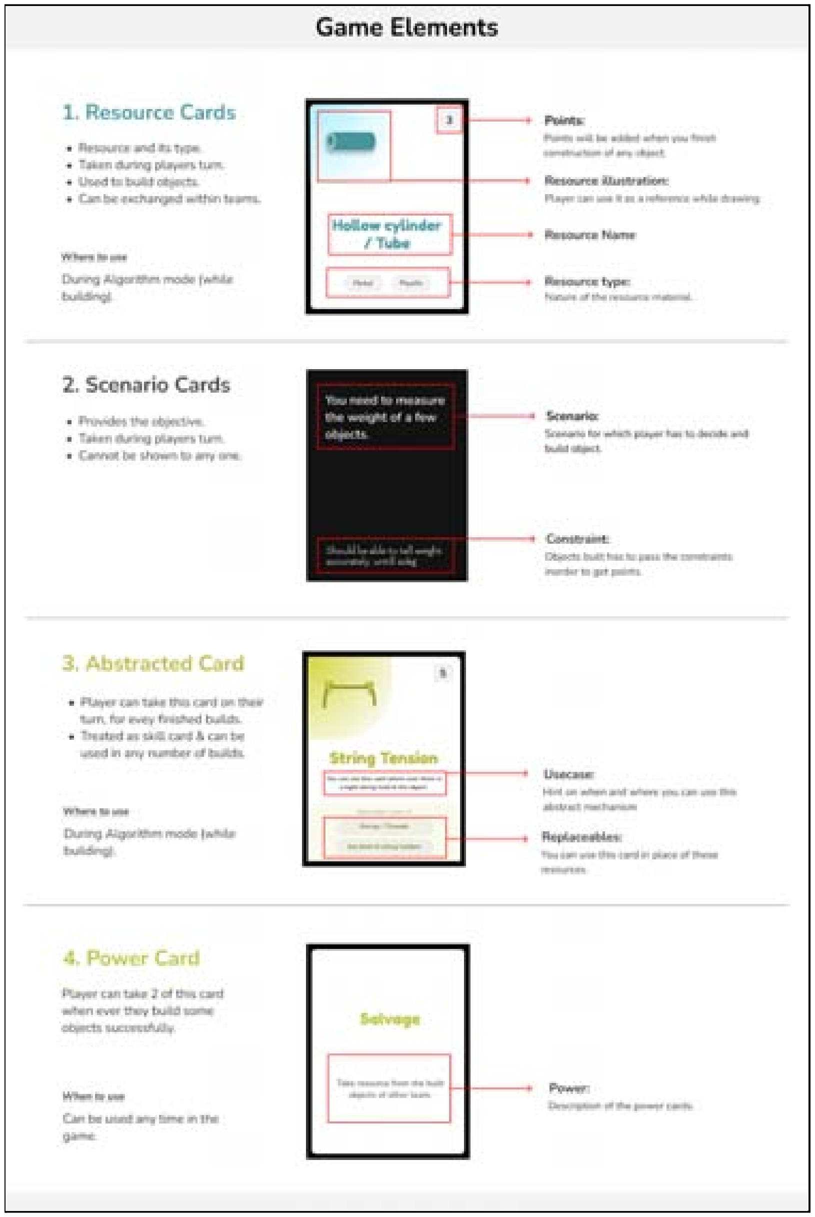
Scenario / buildable Cards:
Scenario that the player has to tackle, for which the player needs to make some objects, as shown in figure 39. A single scenario can have multiple ways of solving it. It is up to players on what they want to build.

Resource cards:
Each card has a resource name, type of materials, and points, as shown in figure 39. A plural resource card can be used multiple times in the same building. A singular resource card should be used only once during construction.

Abstracted function Cards:
These are mechanism cards that are abstractions of some working function, also it is made up of various resources. Players can use this card in place of multiple resources. These are treated as skill cards.

Power card:
To make the game fun, there are few power cards like steal, prevent, skip, create your resource etc. Players can use it any time in the game.

Construction Sheet: It is a white paper which has grids and instruction guides on which players draw to construct objects, as shown in figure 40.



1. Resource Cards
- Resource and its type.
- Taken during players turn.
- Used to build objects.
- Can be exchanged within teams.

When to use
During Algorithm mode (while building).

Points: Points will be added when you finish construction of any object.
Resource Illustration: Player can use it as a reference while drawing.
Resource type: Nature of the resource material.

2. Scenario Cards
- Provides the objective.
- Taken during players turn.
- Cannot be shown to any one.

You need to measure the weight of a few objects.
Should be able to tell weight accurately, until 10kg.

Scenario: Scenario for which player has to decide and build object.
Constraint: Objects built has to pass the constraints inorder to get points.

3. Abstracted Card
- Player can take this card on their turn, for evey finished builds.
- Treated as skill card & can be used in any number of builds.

Where to use
During Algorithm mode (while building).

Usecase: Hint on when and where you can use this abstract mechanism
Replaceables: You can use this card in place of these resources.

4. Power Card
Player can take 2 of this card when ever they build some objects successfully.

When to use
Can be used any time in the game.

Take resource from the built objects of other team.

Power: Description of the power cards.

Fig 39. Explanation of game elements.

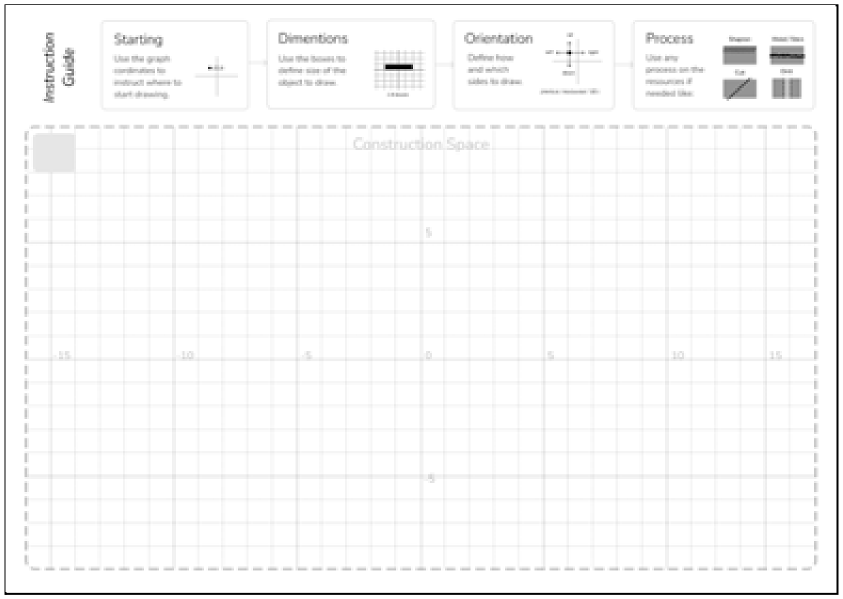


Fig 40. Construction sheet, in which players draw.

Game setting

Initial game settings were shown in figures 41 and 42.

- Four-player game with two people on each team.
- 16 resource cards open up on the table and it gets replenished as people draw it.
- Abstracted function cards will be arranged in the slots.
- The scenario cards, Power cards, lies upside down.
- Construction sheets are given to players when they wish to build.

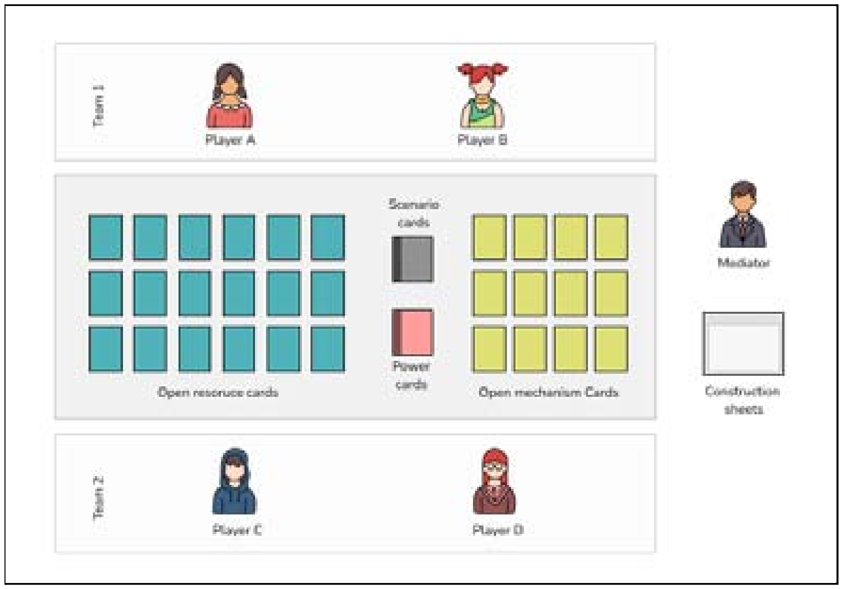


Fig 41. Explanation of game settings.

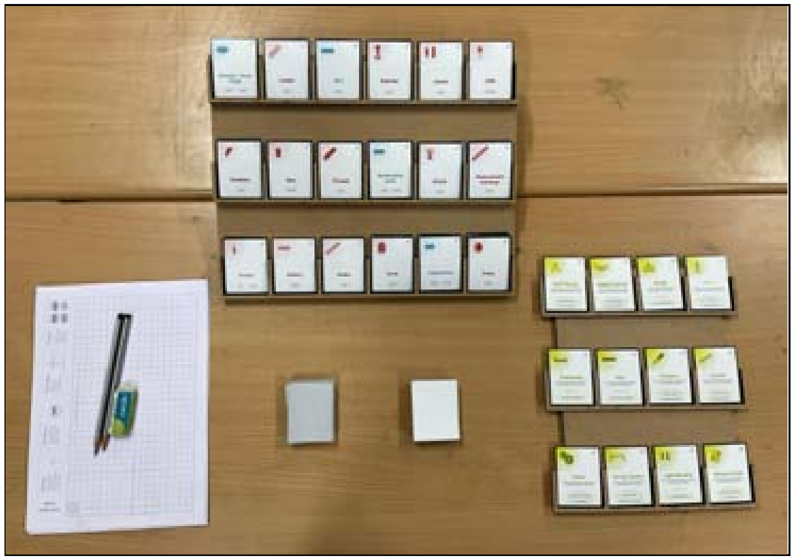


Fig 42. Excel sheet containing the content of the game.

Player Actions

- In each turn a player can either:
    - Take a maximum of two scenario cards.
    - Take a maximum of 3 resource cards.
    - Go into building mode (collaborate and build).
    - Go into upgrade mode (to finish the existing build).
    - Exchange resource cards with teammates.
    - Take 1 abstracted card (only after 1st successful build).

- After a successful build. The player gets the respective points, can take three power cards, and gets access to take abstracted cards.

- Power cards can be used any time in the game.

- Team members cannot reveal scenario cards to other team members. Only resource cards can be revealed to teammates.

Gaining Points

Every time a team successfully builds something. The team gets the points from resource cards used to build the specific object. The points accumulate each time a team builds a new object.

Winning conditions

The game lasts several rounds, and the player address many scenarios. At last, the winning conditions are determined by:

- **Point-based:** The first team to get more than 30 points, wins the game.

- **Time-based:** The one who scores the maximum point within the time wins the game.

The game flow starts from here.

This game has two part. First is resource collection, where players use decomposition, pattern recognition and abstraction.
The 2nd part is the building (algorithm mode), where the player uses algorithms and abstraction.

Resource collection:

Game flow:

- The player first picks up the scenario card.
- Then slowly, the player picks the related resource card from the pile of resource cards, which will help them tackle the scenarios.
- If they run out of related resources, they can pick a new scenario card anytime.
- Players can discuss and exchange resource cards with their teammates.
- Once they collected enough resource cards to address the scenario.
- The team can go to Algorithm mode (building mode).

Algorithm mode (building mode)

Once a team has enough resources to build any object. They can put down the specific scenario card upside down and shout, “I am going to algorithm mode”.

In building mode, the team gets a construction sheet, and they have to draw to construct an object.

In algorithm mode, roles will be assigned:

Active team: Instructor, Builder

Opposite team: Supervisors

Instructor: a player who starts the building process they instruct a teammate to build.

Builder: another teammate acts like a robot and executes (draws) whatever the instructor says.

Supervisor: opposite team members who monitor if the builder mis-performs the instruction.

Game flow:

- The instructor can start to instruct his teammate (builder) to start building the objects
- Instructors can use the instruction guide to form sentences.
- The builder has to draw whatever the instructor says, on the construction sheet.
- Supervisors will be monitoring if the builder does something apart from what the instructor says and if they find any, they can stop the construction.
- After the building is done. Building reveal the scenario card.
- Supervisors will judge the built object, and have to agree upon the functionality of the object.
- Building team can justify it. If they succeed, they win the points from the used resource cards

Scaffolding for providing Instruction in algorithm mode

- Use Object card: (which resource card you are using)
- Define Size: (what size you want to draw the object)
- Placement: (where do you instruct to place the object)
- Alignment: (how do you align the placed object to others)
- Process: (you want to do any process to the objects, like drill a hole from the side)

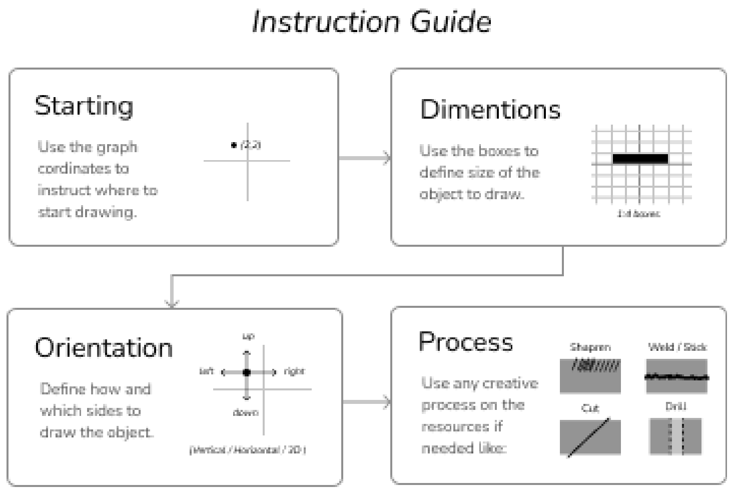

Instruction Guide
Starting
Use the graph cordinates to instruct where to start drawing.
(2,2)
Dimentions
Use the boxes to define size of the object to draw.
1:4 boxes
Orientation
Define how and which sides to draw the object.
up
left
right
down
(Vertical / Horizontal / 3D)
Process
Use any creative process on the resources if needed like:
Sharpen
Weld / Stick
Cut
Drill

Fig 43. Instruction guide in construction sheet.

Previous Iterations

Before the final prototype, the paper prototype was build to test various iterations as shown in figure 44, three to four iterations of design changes have been done to the paper prototype, paper prototype helps in quickly changing rules and dynamics to evaluate the iterations.

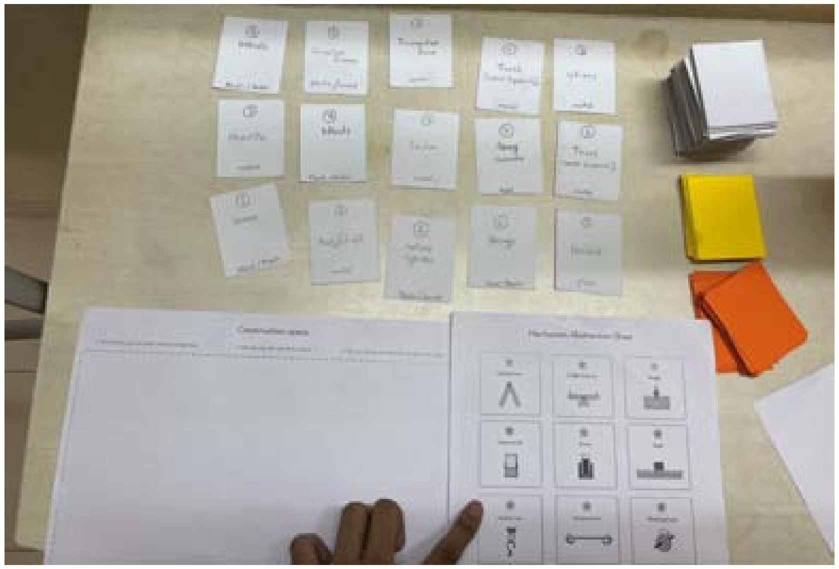


Fig 44. A previous iteration of the paper prototypes.

Results of Games

The compilation of objects that players built in the various game play testing is shown in figure 45.

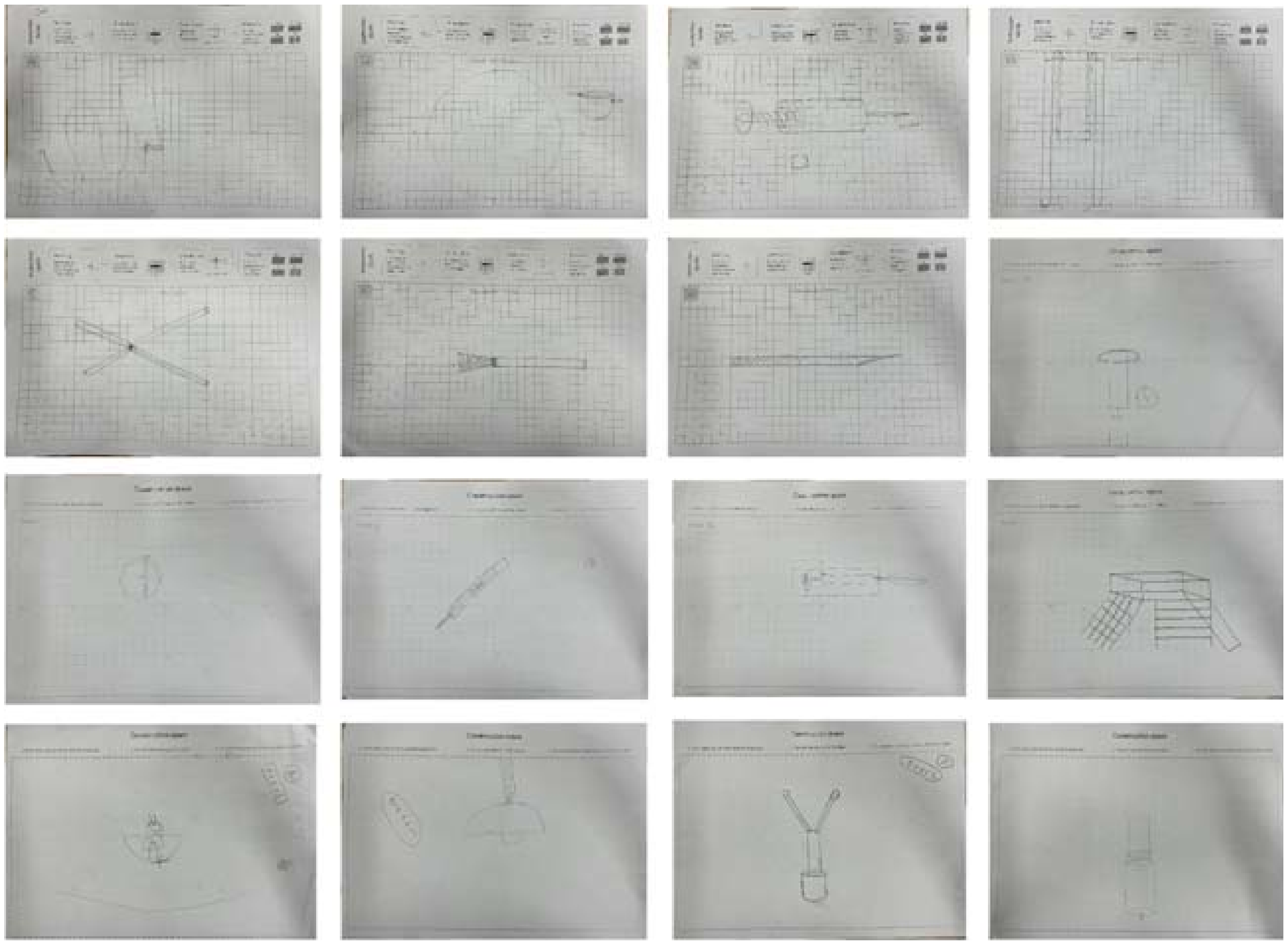


Fig 45. Objects built in the game play testing.

Gameplay from the player's point of view

This section details what players were thinking during the gameplay in each stage and how it contributes to the learning of computational thinking. The text mentioned in the blue colour are examples.

1. **When they pick scenario cards.**

They would be thinking about various possible solutions to address the scenario.

Say for example a scenario card which says “you are responsible for planning a children's playground in school. What will you build?”

Now the player has to think about what kind of items will be in the playground. Like a swing, seesaw, slides, merry go round, etc. Now he had decomposed the scenario into many possible directions. And after choosing one, if they chose to build a swing, they would be decomposing swing to think about materials that make up swing (chains, rod, seats etc).

2. **When they are looking at the resource cards.**

After deciding on what they wanted to build, they would have a list down resources needed to build the object in their head. When they were looking at the resource cards they would be cross verifying if it was there and they would try to pick those cards.

Sometimes they will be looking at the resource cards in the deck and back trace if this resource would help them to build that specific object they choose.

Say for example: If the player decides to build a swing. He already has a rod as a resource but the resources deck doesn't have a chain or string needed for that, but other resources like screws and seats are present in the deck. Now the player can change the object according to the materials and he can choose to build a sea-saw instead of a swing.

Here is when they branch their thinking, if this resource is available they would choose one object to build or if that was not available they would go for another other object. So they were continuously branching their thoughts.

In both ways, they were trying to decompose the object into smaller pieces to pick resources.

3. **When they are looking at other picking the resources**

During others' turn, when others were picking the resource cards, the player would be trying to guess other player’s scenario card based on the resources they picked. By analyzing the pattern of resource cards other player picks, and understanding the relationship between the resources they picked.

Say for example: When others were picking resources like battery, wire, switch etc. they could easily analyze the pattern and determine what object others would possibly build, that it could be something to do with the electric circuit, it could be torch, bell etc..

4. **When they are in Algorithm mode.**

When Instructing the player has to think about the sequence of steps and in which they have to unfold resources, and also think about how to deliver the instruction precisely by using the instruction guide. Players can get creative with the recourse and use any kind of process, so they will be thinking about various ways to build.

For example: When they were instructed to build a swing, they first looked at various resources in their hand and they instructed the teammate to cut the rods into various pieces. And mention the dimensions and then build the frame, then attach the chains to it, then add the seat and then weld them together.

5. **When they are looking at Abstracted function cards .**

When they look at the abstract card they see if the functions/mechanism present in the card is a part of the object that they are trying to build. So they might pick the mechanism and use it in the object instead of picking many resources from the resources deck.

Say for example: if they wanted to build scissors for the specific scenario card, instead of going for resources like rectangular pieces, holders, screws and making the fulcrum mechanism They can directly pick the middle fulcrum mechanism card from the abstracted cards deck and build scissor with it.

Where and when they learn CT in the gameplay

There are several places in the game where the players learn about Computational thinking either knowingly or unknowingly, those are listed below.

Decomposition:

- When they analyse the scenario card and find multiple ways to solve the problem.
- Breaking down the objects and thinking about the parts that make them.

Pattern recognition:

- recognizing the draw pattern of the teammate.
- recognizing the draw pattern of the opposite team.
- recognizing which resource to pick first based on decomposed objects.
- Constantly thinking about what resource to pick.
- Recognizing the pattern to which the builder responds when he is drawing and adapting to a style of instruction.

Algorithm:

- During building mode they think about how to instruct their teammate, thinking in sequential order about what comes first and next.
- Since the builder acts like a robot. precise instruction has to be given. The precision of instruction by considering all possible mistakes a builder can make just like how a computer does.

Abstraction:

- And when they are trying to use the abstracted function card while making new objects

Other learnings:

- Collaboration Skill
- Communication Skill
- Manufacturing process
- Mechanisms

9. Evaluation

Evaluation method

In the Initial stages, various types of evaluation were carried out. With direct subjective questions about computational thinking, questionnaire with various examples of computational thinking and game engagement questionnaires.

Later after a few iterations of the game, a new evaluation plan was made in which game quality was measured along with some qualitative findings.

For finding game quality MEEGA+ (A Method for the Evaluation of Educational Games for Computing Education) [17] was used. For qualitative analysis, think-aloud experiments in various stages of game play were performed, and Think-aloud analysis was carried out for a post-test where players were asked to solve some hypothetical problems.

Game testing evaluations were carried out with 10th school students from IIT KV for each iteration of the game and also a few rounds of testing with our batchmates to get more insights from thinking aloud evaluation.

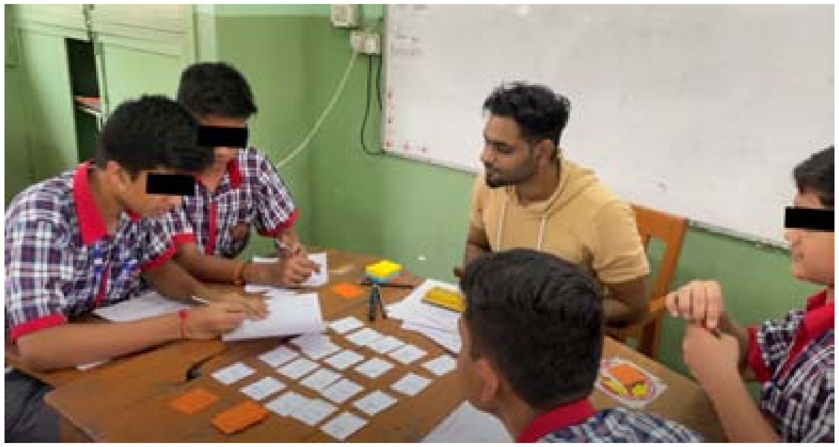


Fig 46. Game play testing session in KV school.

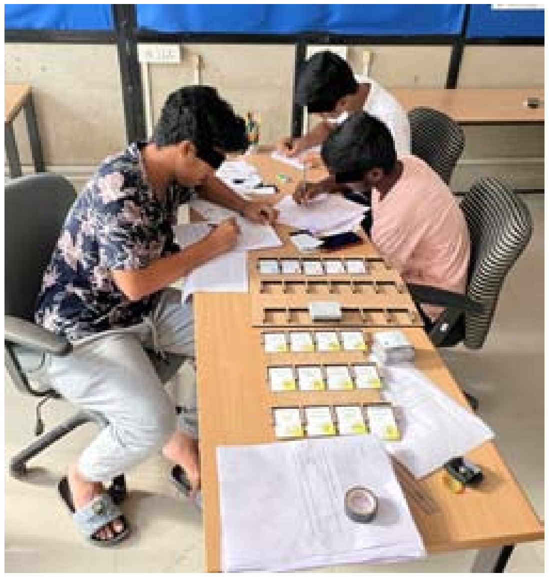


Fig 47. Game play testing of the final prototype.

Game quality evaluation using the MEEGA+ model

The Excel sheet with questionnaire data.

MEEGA+ is a systematic model to analyse educational games (digital and nondigital ones) [17] to evaluate their perceived quality from the student's perspective in the context of computing education.
It consists of a list of a questionnaire which was answered on a 5-point Likert scale with response alternatives ranging from strongly disagree to strongly agree.

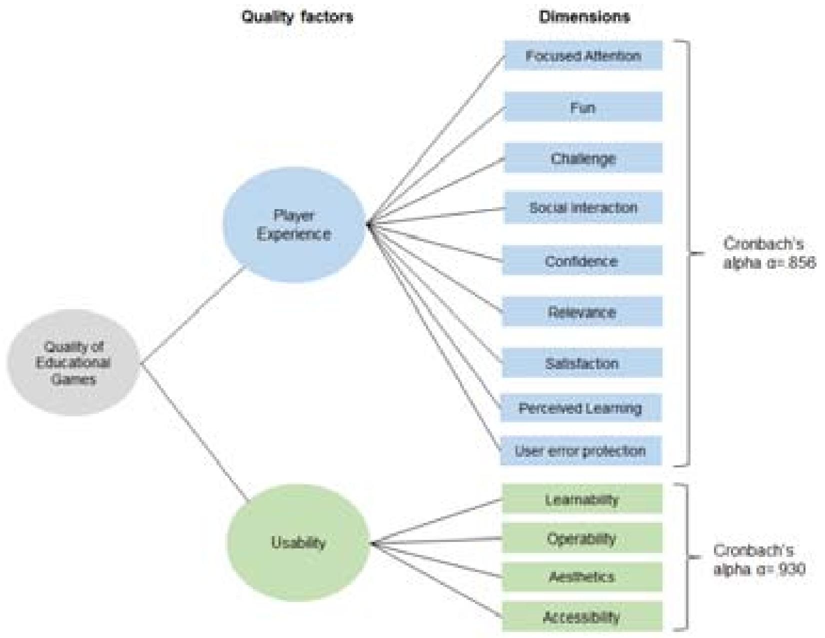


Fig 48. Dimension in MEEGA+ evaluation of the model.

MEEGA questionnaire covers the dimensions like useability, confidence, challenge, satisfaction, social interaction, fun, focused attention, relevance (content), and perceived learning. Also, the Cronbach's alpha value of the questionnaire is above 0.9 which indicates the reliability of the questionnaire.

Dimension/Sub dimension		Item No.	Description
Usability	Aesthetics	1	The game design is attractive (interface, graphics, cards, boards, etc.)
		2	The text font and colors are well blended and consistent
	Learnability	3	I needed to learn a few things before I could play the game
		4	Learning to play this game was easy for me
		5	I think that most people would learn to play this game very quickly
	Operability	6	I think that the game is easy to play
		7	The game rules are clear and easy to understand
	Accessibility	8	The fonts (size and style) used in the game are easy to read
		9	The colors used in the game are clear and meaningful
	User error protection	10	The game prevents me from making mistakes
		11	When I make a mistake, it is easy to recover from it quickly
Confidence		12	When I first looked at the game, I had the impression that it would be easy for me

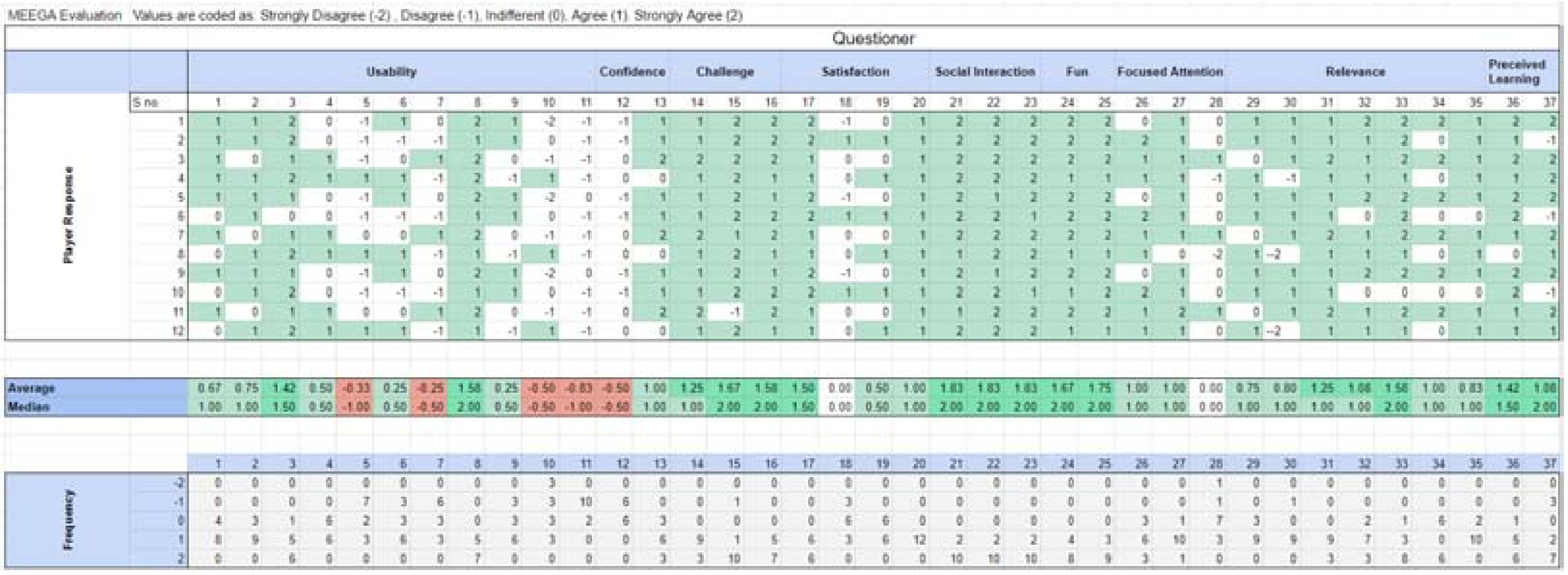
	13	The contents and structure helped me to become confident that I would learn with this game
Challenge	14	This game is appropriately challenging for me
	15	The game provides new challenges (offers new obstacles, situations, or variations) at an appropriate pace
	16	The game does not become monotonous as it progresses (repetitive or boring tasks)
Satisfaction	17	Completing the game tasks gave me a satisfying feeling of accomplishment
	18	It is due to my personal effort that I managed to advance in the game
	19	I feel satisfied with the things that I learned from the game
	20	I would recommend this game to my colleagues
Social interaction	21	I was able to interact with other players during the game
	22	The game promotes cooperation and/or competition among the players
	23	I felt good interacting with other players during the game
Fun	24	I had fun with the game
	25	Something happened during the game (game elements, competition, etc.) which made me smile
Focused attention	26	There was something interesting at the beginning of the game that captured my attention
	27	I was so involved in my gaming task that I lost track of time
	28	I forgot about my immediate surroundings while playing this game
Relevance	29	The game contents are relevant to my interests
	30	It is clear to me how the contents of the game make me identify some patterns
	31	It is clear to me how the contents of the game make me understand general mechanisms
	32	It is clear to me how the contents of the game make me think about breaking down the objects
	33	It is clear to me how the contents of the game make me think of instructions step by step.
	34	This game is an adequate teaching method for thinking skill
	35	I prefer learning with this game to learning through other ways
Perceived learning	36	The game contributed to my learning in problem-solving in some scenarios
	37	The game allowed for efficient learning compared with other activities in the school

Table 6. MEEGA questionnaire.

Around three game playtesting sessions were conducted with 4 people in each gameplay. Which leads to the 12 entries. Later the data were coded as: Strongly Disagree (-2) , Disagree (-1), Indifferent (0), and Agree (1). Strongly Agree (2). Average, median and frequency are calculated to draw insights.

Quantitative results for measuring game quality

The data were coded & tabulated in an excel sheet to analyze as shown in table 7. Later mean median, and frequency was calculated, and visualized in the graph as shown in figure 49 and 50



MEEGA Evaluation Values are coded as: Strongly Disagree (-2), Disagree (-1), Indifferent (0), Agree (1), Strongly Agree (2)

Questioner

		Usability											Confidence		Challenge			Satisfaction	
	S no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Player Response	1	1	1	2	0	-1	1	0	2	1	-2	-1	-1	1	1	2	2	2	-1	0
	2	1	1	2	0	-1	-1	-1	1	1	0	-1	-1	1	1	2	2	2	1	1
	3	1	0	1	1	-1	0	1	2	0	-1	-1	0	2	2	2	2	1	0	0
	4	1	1	2	1	1	1	-1	2	-1	1	-1	0	0	1	2	1	1	0	1
	5	1	1	1	0	-1	1	0	2	1	-2	0	-1	1	1	2	1	2	-1	0
	6	0	1	0	0	-1	-1	-1	1	1	0	-1	-1	1	1	2	2	2	1	1
	7	1	0	1	1	0	0	1	2	0	-1	-1	0	2	2	1	2	1	0	0
	8	0	1	2	1	1	1	-1	1	-1	1	-1	0	0	1	2	1	1	0	1
	9	1	1	1	0	-1	1	0	2	1	-2	0	-1	1	1	2	1	2	-1	0
	10	0	1	2	0	-1	-1	-1	1	1	0	-1	-1	1	1	2	2	2	1	1
	11	1	0	1	1	0	0	1	2	0	-1	-1	0	2	2	-1	2	1	0	0
	12	0	1	2	1	1	1	-1	1	-1	1	-1	0	0	1	2	1	1	0	1
Average		0.67	0.75	1.42	0.50	-0.33	0.25	-0.25	1.58	0.25	-0.50	-0.83	-0.50	1.00	1.25	1.67	1.58	1.50	0.00	0.50
Median		1.00	1.00	1.50	0.50	-1.00	0.50	-0.50	2.00	0.50	-0.50	-1.00	-0.50	1.00	1.00	2.00	2.00	1.50	0.00	0.50

		Satisfaction	Social Interaction			Fun		Focused Attention			Relevance							Preceived Learning	
	S no.	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Player Response	1	1	2	2	2	2	2	0	1	0	1	1	1	2	2	2	1	2	2
	2	1	2	2	2	2	2	2	1	0	1	1	1	1	2	0	1	1	-1
	3	1	2	2	2	2	2	1	1	1	0	1	2	1	2	2	1	2	2
	4	1	2	2	2	1	1	1	1	-1	1	-1	1	1	1	0	1	1	2
	5	1	2	1	2	2	2	0	1	0	1	1	1	2	2	2	1	2	2
	6	1	2	2	1	2	2	2	1	0	1	1	1	0	2	0	0	2	-1
	7	1	2	2	2	2	2	1	1	1	0	1	2	1	2	2	1	1	2
	8	1	1	2	2	1	1	1	0	-2	1	--2	1	1	1	0	1	0	1
	9	1	2	1	2	2	2	0	1	0	1	1	1	2	2	2	1	2	2
	10	1	2	2	1	1	2	2	1	0	1	1	1	0	0	0	0	2	-1
	11	1	1	2	2	2	2	1	2	1	0	1	2	1	2	2	1	1	2
	12	1	2	2	2	1	1	1	1	0	1	--2	1	1	1	0	1	1	1
Average		1.00	1.83	1.83	1.83	1.67	1.75	1.00	1.00	0.00	0.75	0.80	1.25	1.08	1.58	1.00	0.83	1.42	1.08
Median		1.00	2.00	2.00	2.00	2.00	2.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.50	2.00

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Frequency	-2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
	-1	0	0	0	0	7	3	6	0	3	3	10	6	0	0	1	0	0	3	0
	0	4	3	1	6	2	3	3	0	3	3	2	6	3	0	0	0	0	6	6
	1	8	9	5	6	3	6	3	5	6	3	0	0	6	9	1	5	6	3	6
	2	0	0	6	0	0	0	0	7	0	0	0	0	3	3	10	7	6	0	0

		20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Frequency	-2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	-1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	3
	0	0	0	0	0	0	0	3	1	7	3	0	0	2	1	6	2	1	0
	1	12	2	2	2	4	3	6	10	3	9	9	9	7	3	0	10	5	2
	2	0	10	10	10	8	9	3	1	0	0	0	3	3	8	6	0	6	7

Table 7. Data collection of the MEEGA questionnaire.

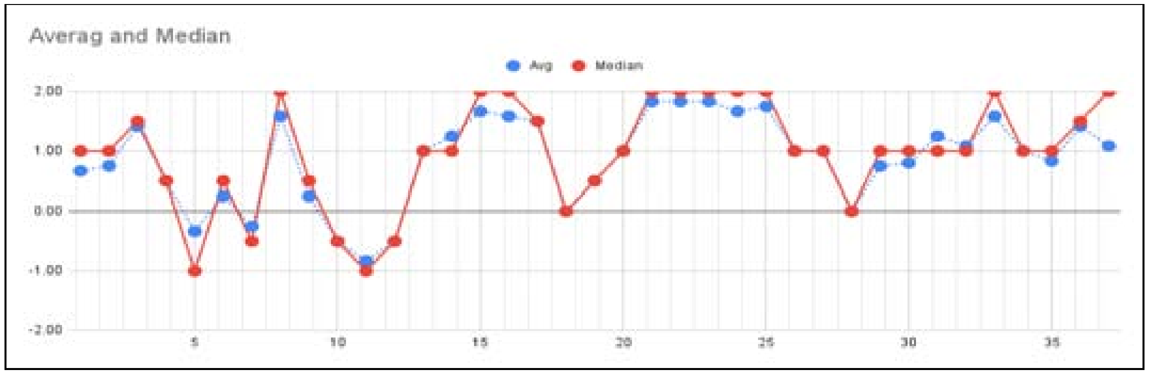


Fig 49. Mean and median graph of the MEEGA questionnaire.

Inference

Questions which got bad scores, whose median is below 0,

- 5. I think that most people would learn to play this game very quickly.
- 7. The game rules are clear and easy to understand.
- 10. The game prevents me from making mistakes.
- 11. When I make a mistake, it is easy to recover from it quickly.
- 12. When I first looked at the game, I had the impression that it would be easy for me.

It indicates that the game is performing poorly in sub-dimension of useability like “error prevention”, some aspects of “learnability of the game”, and “first impression on ease of play”.

For questions which were indifferent, the median is 0, which means not much insight can be drawn from these questions.

- 18. It is due to my personal effort that I managed to advance in the game. (it could be combined team effort which is required in the game)
- 28. I forgot about my immediate surroundings while playing this game, (but the other two dimensions in focused attention got scores above one).

Apart from these questions, all other dimensions of games were performing well and had a median above one. Dimensions like a challenge, social interaction, fun, focused attention, satisfaction, relevance, and perceived learning have good scores. Quantitative data shows that the overall game seems to be performing well.

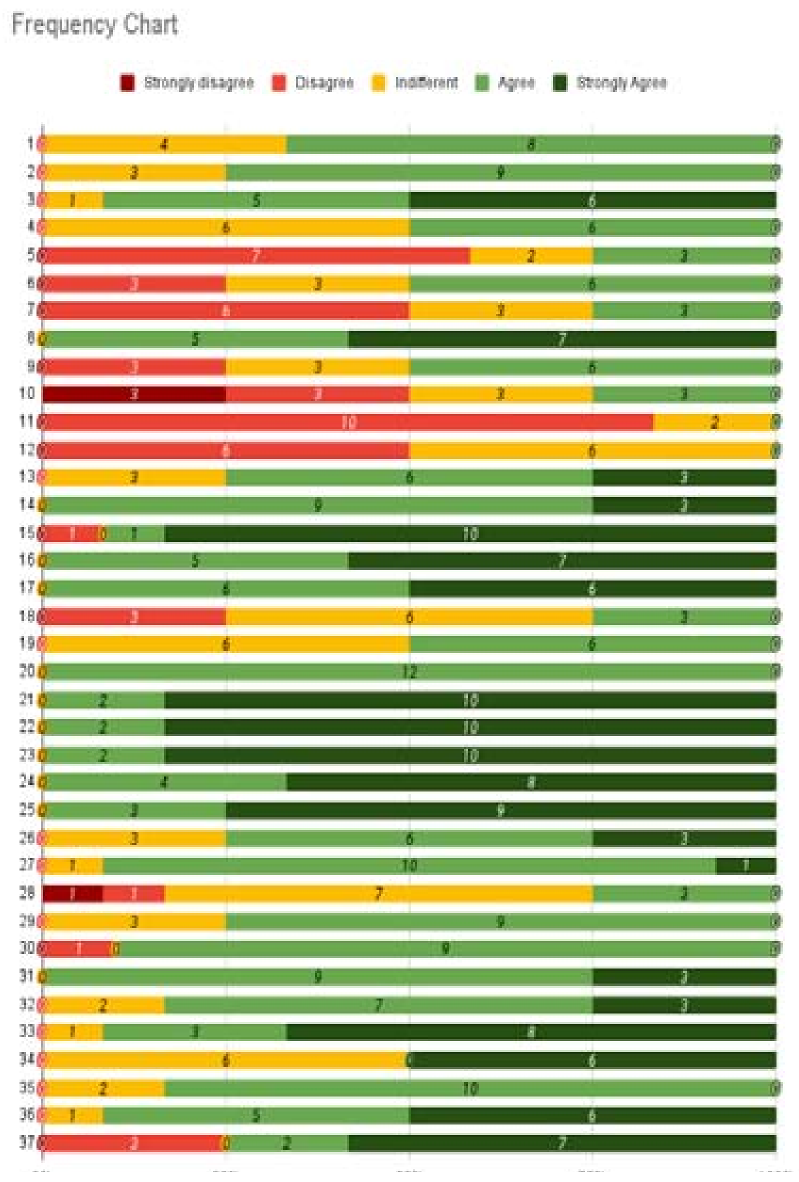


Fig 50. Frequency chart of the MEEGA questionnaire.

Qualitative analysis

Observations:

By notes taken during the gameplay and also by analysing the recorded video, observations are presented in the below section.

Previous iterations:

- Players were keen on building the objects.
- The players didn't make objects according to my expectations. They made some novel objects for the same scenarios with whatever resources they had.
- There was always a moment of surprise for everyone when they finished the construction of any object.
- The game's implementation of decomposition and algorithm seems to be clearly visible during gameplay, but they didn't use mechanism abstraction in the game.
- There were lots of discussions during an exchange of cards, and good team dynamics were found.
- During the justification of the objects made by teams, surprisingly there were no conflicts, and a good inter-team interaction was found.
- They used erasers a lot initially when drawing, which indicates the instructions given were not accurate.

Final iteration:

- "Steal power cards have to be removed since players were keen on noticing what resource cards others were drawing with the intention to steal rather than the intention to figure out what other team is going to build.
- The opposite team doesn't tend to supervise the building action as expected. Most of the time they were distracted by their own resources and they pay attention to the building, more during the start and at the end of the building.
- People reported error prevention was not there in the game, when questioned why? They mentioned it's hard to prevent errors, especially in building mode.
- They were picking resources with wide or vast applications when none of the resources are available for the specific scenario cards.
- Players rarely used the abstarted function cards, and have to make them use it more to realize how abstraction works.
- Players didnt use much scenario cards.

Strong aspects of the game

According to the survey results, the strong aspects of the game are:

- The game was engaging and forces one to be creative to solve problems, and also makes you better at giving detailed instructions.

- It forces thinking, learning about building mechanisms, product design, and breaking down objects to small elements.

- Teaches problem-solving in a step by step approach.

- Makes peoople think about various processes, how to break down in smaller steps, and how to communicate instructions.

- The game idea was excellent. The game has made me creatively think more than usual.

Learning evaluation

In order to determine if students were able to grasp concepts of computational thinking, think-aloud analysis was carried out after the game play.

Think aloud analysis on gameplay

After the gameplay was done, think aloud analysis was performed with all the players and recorded their voice.
Interview protocol was set with the base questions, and probed them more whenever required.

Think aloud evaluation was performed for all the possible actions in the game. The results were recorded, and later interpreted.

The text in **blue are the questions** asked and the text in **green are inferences** from the think-aloud analysis.

1. *When asked, "what were you thinking when you had a scenario card in your hand?"*

   Most of them replied:

   "I was thinking about the list of objects to make and all the resources needed to make the object required for the scenario"

   Some of them replied:

   "I will start picturing an overview of the object, then I look at my resource cards again. And try thinking or fitting the object to make a product, or go back and forth until I come up with a product."

   "I didn't notice the constraints in the that much since I was into thinking about the objects"

   It is evident that almost all the players who played the game were doing decomposition for the objects required to build.

2. *"What were you thinking when you were looking at the resource cards in the deck and when picking it?"*

   Most of the players replied:

   "What resources cards can be used to complete my scenario"

   "Specifically looking for the resource cards which will be helpful to build the object I have in mind"

   Some of them replied:

   "Can I make any other new object with the resource in the resource deck"

   "What resource could be tweaked the most, which have a wide or vast application, I would go for it if I don't have anything relevant to my scenario card"

   It is evident that they already have a list in their mind and they are looking for the resources in the deck, which means the breakdown of scenario and object has already happened in their head.

3. *What were you thinking when others were picking the resource card?*

<u>*Most of the players replied:*</u>

"I was trying to remember all the resource card which others are picking"

"I was keeping track of what other players picking inorder guess what they are building"

"I didn't want them to pick resource cards which are necessary for me"

"How to obstruct the other player, based on what they pick"

<u>*Some players replied:*</u>

"That they are keeping an eye on what other are picking so that they can steal the cards if the resource is required for them"

"I was curious about what is beneath the current resource card when other pick the top card"

It is evident that they were clear about keeping track of what resources others were picking. Also, they were trying to guess the pattern, based on the sequence and type of resource the other teams were picking, in order to figure out others build.

4. *What are your thoughts when you are instructing your teammate?*

<u>*Most of the players replied:*</u>

"I was extra careful not to use eraser while building(Drawing), since the use of eraser has negative points"

"How to instruct as clear and precise as possible"

"How to use the grids ,boxes and processes in the sheet"

"That I have to instruct in the way teammate would understand and draw it"

<u>*Some players replied*</u>

"Was thinking, with what resource should I stared building and what resource to use next next"

"I was trying to frame the instructions in our common language since the teammate is my friend"

"How to make the object look as convincing as possible for the other team"

It is evident that they were clear about giving precise instructions. And they were doing sequential thinking to create steps required to instruct.

5. *What were you thinking when you were looking at the abstracted cards?*

Most of the players replied:

"Initially was curious, confused and little intimidated, after looking at visuals and the card for longer, it made sense"

"What are the resources I could replace one of those cards with."

"Does the object I am trying to build have any mechanism in it, and if I can use these cards?"

Few players did not answer this question at all

It is clear that some of them didn't understand the abstraction in detail. Maybe it's because of the cards or less importance given to the functions cards in the game. However, people were able to think about if the abstracted functions are applicable to their objects.

6. *Have you changed the object you wanted to build in between the game? If so, why?*

Most of the players replied:

"When new resources get revealed I always realized there are alternative ways of building the objects for my scenario card"

"Yes, I was thinking what other alternative ways to do, how to cut down the instructions with new resources"

"I changed because I wanted to reduce the instructions steps, to reduce the error"

Few players replied:

"To increase my points, I was adding more resources to detail the built"

"I would ratherdetails the same build with more resources instead of thinking about new scenario/object to build"

"No I have not changed in between, somehow I tried to build same object I have in mind".

It is clear that they were dynamically changing some of the decisions in the game based on what resources were available to them at that point of the time. They are using if conditions in their thinking, like, If this resource is available, go for building this object or go for another object.

Improvements

Game improvements from previous iterations.

- Improved visuals and added illustrations to the cards to make them easily recognise the resources at a glance and also it acts as a guid during drawing action.
- Change in placement of illustrations from center to top left corner. Generally people hold cards in a particular fashion which reveals only the top left position of the card.
- Mapping multiple objects to a single scenario gives them creative freedom to switch between objects while collecting resources.
- Adding more primitive shapes since players were using primitive shapes more in the game.
- Including the terms of computational thinking in the game so that verbal assurance helps them associate the action with the word. For example, every time they are going to build something, they have to yell “entering algorithm mode”
- Improved instruction guide, in a visual format, which provides them scaffolding while instructing others.
- Introduced guides, grids in the construction sheet to facilitate drawing action.

Improvements after final testing

To improve error prevention:

- Allowing finger pointing action in the construction sheet helps them to correct the instructions in case it goes wrong.
- A visual rule book with scenarios could help them easily recognize rules in case of players during instruction.
- Increasing the font size, and contrast in the abstracted function cards.
- Switching to standard card size instead of mini cards, will have more space for contents in the cards.

To improve ease of access:

- Providing various shapes for different types of cards will help them instantly recognize the cards.
- Switching to standard card size instead of mini cards.

To improve learnings of abstraction:

- People were motivated to attain power cards in the game. This can be used as an incentive to trigger them to use abstract function cards more in the game. Eg: if everytime they pick or use an abstracted function card or use an abstracted function card, they will get one power card.

General improvements

- Some of the power cards are very distractive, like stealing resource cards, which makes the players focus on only stealing other’s resources instead of making them guess what others are going to build, so it has to be removed. Power cards that do not affect learning can be present in games like play twice, skip, prevent, etc.

10. Conclusion

We made the game after analyzing various game design frameworks and various board games, which helped us ideate and develop a proper context for the game to teach computational thinking. Later we developed a paper prototype and tested it with lots of playtesting and iterations to mature the game. Finally, the evaluation was done to figure out the game quality and learning outcomes.
It was found that the game was challenging, fun, and exciting to play. It also pushed players to think creatively along with a lot of social interactions, although people lost focus after one hour and thirty minutes into gameplay.

Regarding learning evaluation, players were able to grasp the concepts of decomposition, algorithm and pattern recognition very clearly, strongly and explicitly. But they were only able to understand the concepts of abstraction implicitly.
We conclude that the game has enough potential to teach computational thinking in a fun and exciting way.

Personal learnings

I was able to play multiple board games in a short time and understand the dynamics of various games, it was fun and engaging, and it helped me to come up with ideas for designing this game. I also started applying computational thinking in my design process whenever it is required, in fact, we used computational thinking to ideate for this project.
The game design approach was quite different from normal design thinking. It demanded more playtesting and iterations. We were able to do a lot of thought experiments where we were initially visualizing the whole gameplay to test how minor changes in-game rules affect the whole gameplay.

11. References

Publications

1. Piaget, J. (1964). **Cognitive development in children**: Piaget.Journal of research in science teaching, 2(3), 176-186.
2. Piaget, J. (2013). **Play, dreams and imitation in childhood** (Vol. 25). Routledge.
3. Kudrowitz, Barry & Wallace, David. (2010). **The play pyramid: A play classification and ideation tool for toy design**. Int. J. Arts and Technology. 3. 10.1504/IJART.2010.030492.
4. Cynthia C. Selby. 2015. **Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy.** In Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15). Association for Computing Machinery, New York, NY, USA, 80–87. DOI:https://doi.org/10.1145/2818314.2818315
5. Ting-Chia Hsu, Shao-Chen Chang, Yu-Ting Hung, **How to learn and how to teach computational thinking: Suggestions based on a review of the literature**, Computers & Education, Volume 126, 2018, Pages 296-310, ISSN 0360-1315, https://doi.org/10.1016/j.compedu.2018.07.004.
6. Basawapatna, Ashok & Kyu, Han & Koh, Kyu Han & Repenning, Alexander & Webb, David & Marshall, Krista. (2011). **Recognizing computational thinking patterns**. SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. 10.1145/1953163.1953241.
7. Prensky, M.: Digital game-based learning. McGraw-Hill, New York (2001)
8. https://www.thetech.org/sites/default/files/techtip_computationalthinking_v3.pdf
9. "Developing Computational Thinking in Compulsory Education" by the Joint Research Centre (JRC), the European Commission's science and knowledge service. https://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188_computhinkreport.pdf
10. Dsource: Designing for Children - Play and Learn: Caillois's Attitudes in Play Experience https://www.dsource.in/course/designing-children-play-and-learn/play-theories-and-design/csikszentmihalyi%E2%80%99s-flow-theory
11. Dsource: Designing for Children - Play and Learn: Csikszentmihalyi's Flow Theory https://www.dsource.in/course/designing-children-play-and-learn/play-theories-and-design/caillois%E2%80%99s-attitudes-play
12. Dsource: Designing for Children - Play and Learn: Piaget's Theory of Cognitive Development https://www.dsource.in/course/designing-children-play-and-learn/learning-theories-and-design/piaget%E2%80%99s-theory-cognitive
13. Kelly, N., & Gero, J. (2021). Design thinking and computational thinking: A dual process model for addressing design problems. Design Science, 7, E8. doi:10.1017/dsj.2021.7
14. https://pressbooks.library.ryerson.ca/guide/chapter/1-3-the-art-of-serious-game-design-conceptual-framework/
15. Hunicke, Robin & Leblanc, Marc & Zubek, Robert. (2004). MDA: A Formal Approach to Game Design and Game Research. AAAI Workshop - Technical Report. 1.

https://www.researchgate.net/publication/228884866_MDA_A_Formal_Approach_to_Game_Design_and_Game_Research

16. Ferdig, Richard & Winn, Brian. (2009). The Design, Play, and Experience Framework. 10.4018/978-1-59904-808-6.ch058. https://www.researchgate.net/publication/314280472_The_Design_Play_and_Experience_Framework
17. Petri, Giani & Gresse von Wangenheim, Christiane & Borgatto, Adriano. (2018). MEEGA+: A Method for the Evaluation of Educational Games for Computing Education. https://www.researchgate.net/publication/326722665_MEEGA_A_Method_for_the_Evaluation_of_Educational_Games_for_Computing_Education
18. Tsarava, K., Moeller, K., & Ninaus, M. (2018). Training Computational Thinking through board games: The case of Crabs & Turtles. International Journal of Serious Games, 5(2), 25–44. https://doi.org/10.17083/ijsg.v5i2.248
19. Panagiotis Apostolellis, Michael Stewart, Chris Frisina, and Dennis Kafura. 2014. RaBit EscAPE: a board game for computational thinking. In Proceedings of the 2014 conference on Interaction design and children (IDC '14). Association for Computing Machinery, New York, NY, USA, 349–352. https://doi.org/10.1145/2593968.2610489

Tables

Figures

https://www.dsource.in/course/designing-children-play-and-learn/learning-theories-and-design/piaget%E2%80%99s-theory-cognitive

8. Fig 8. Caillois's Attitudes in Play Experience.
   https://www.dsource.in/course/designing-children-play-and-learn/play-theories-and-design/caillois%E2%80%99s-attitudes-play
9. Fig 9. Graph showing skill level vs challenge level in Flow Theory.
   https://www.dsource.in/course/designing-children-play-and-learn/play-theories-and-design/csikszentmihalyi%E2%80%99s-flow-theory
10. Fig 10. Play Pyramid by Kudrowitz and Wallace.
    Kudrowitz, Barry M. and David R. Wallace. "The play pyramid: a play classification and ideation tool for toy design." Int. J. Arts Technol. 3 (2010): 36-56.
    https://www.semanticscholar.org/paper/The-play-pyramid%3A-a-play-classification-and-tool-Kudrowitz-Wallace/9d2ddf7b5073da301d44b213385dbcd246b82ff6#citing-papers
11. Fig 11. Combined relationship diagram between Bloom's Taxonomy Cognitive Domain, computational thinking skills, and the teaching of programming.
    https://dl.acm.org/doi/10.1145/2818314.2818315
12. Fig 12. MDA (mechanics dynamics and aesthetics) framework.
13. Fig 13. The Art of Serious Game Design by Digital Education Strategies, Ryerson University.
14. Fig 14. DPE framework (Design play and experience)
15. Fig 15. Taco coding App, and tangible elements.
    https://www.playshifu.com/tacto/coding
16. Fig 16. Tangiplay app, along with mini tangible elements.
    https://www.tangiplay.com/
17. Fig 17. Google project bloks, with all its tangible components.
    https://www.i-programmer.info/news/150-training-a-education/9867-google-project-bloks-tangible-programming-for-kids.html
18. Fig 18. Interface of Scratch programming language.
    https://scratch.mit.edu/projects/editor/?tutorial=getStarted
19. Fig 19. Interface of MIT App inventor.
    http://appinventor.mit.edu/
20. Fig 20. Interface of Agentsheets.
    https://agentsheets.com/
21. Fig 21. Game play screens of the game Lightbot.
    https://lightbot.com/
22. Fig 22. Game play screen of games in CodeSpark.
    https://codespark.com/
23. Fig 23. Game play screen of game Algorithm city.
    https://play.google.com/store/apps/details?id=air.MusterenGames.ElHarezmiCoding&hl=en&gl=US
24. Fig 24. Game play screens of game Human resource machine.
    https://tomorrowcorporation.com/humanresourcemachine
25. Fig 25. Game play of Crabs & Turtles
    Tsarava, K., Moeller, K., & Ninaus, M. (2018). Training Computational Thinking through board games: The case of Crabs & Turtles. International Journal of Serious Games, 5(2), 25–44. https://doi.org/10.17083/ijsg.v5i2.248
26. Fig 26. Game play of RaBit EscApe.
    Panagiotis Apostolellis, Michael Stewart, Chris Frisina, and Dennis Kafura. 2014. RaBit EscAPE: a board game for computational thinking. In Proceedings of the 2014 conference on Interaction design and children (IDC '14). Association for Computing Machinery, New York, NY, USA, 349–352. https://doi.org/10.1145/2593968.2610489
27. Fig 27. Various ways in which decomposition, pattern recognition, abstractions, and algorithms can be achieved.