End user programming with personally meaningful objects

Andrew Cyrus Smith
CSIR Meraka Institute and University of South Africa
acsmith@csir.co.za

Helene Gelderblom
University of Pretoria
helene.gelderblom@up.ac.za

Abstract
This project investigated what a tangible programming environment could look like in which the program is an arrangement of personally meaningful objects. We identified Gestalt principles and Semiotic theory to be the theoretic foundations of our project. The Gestalt principles of good continuation and grouping by proximity are particularly relevant to our research. Following the Design Science Research methodology, four iterations each focussed on a different design aspect based on the outcome of the previous iterations. The fifth and final iteration combined learning from the previous designs and introduced the Gestalt principle of grouping by proximity to the programming environment. We concluded the project by deriving a model that reflects the programming environment constructs and the relationships between these.

1. Introduction
Current dominant text and graphic based programming environments such as Java (Eckel, 2006) and Scratch (Resnick et al., 2009) apply the principle of "form follows function". Krippendorff (1989) proposed an alternative approach that he called "form follows meaning" and our research considers what a programming environment could look like in which this is true.

Programming languages are designed by language architects and implemented by software developers. The architect determines what algorithms to include in the language and the developer decides how the algorithms are represented. The user then constructs a program using these representations. In almost all cases, the architect, developer, and user are three distinct persons. The result is that the user is burdened with making sense of the representations chosen by the developer. Our research contemplated what a programming environment could look like in which the user chooses how to represent the language architect's algorithms. In searching for an answer, we encountered Semiotic theory and Gestalt principles and found that these helped solve our problem.

We developed five instantiations of a tangible programming environment with each addressing either a problem identified in the preceding iteration or a new concept that would get us closer to a solution. Our research concluded with a model of a tangible programing environment that explicitly incorporates Gestalt principles and Semiotic theory.

The rest of the paper is structured as follows: Section 2 provides the theoretical background that underpins this research with these being Gestalt principles and Semiotic theory. The methodology we followed is described in Section 3 and Section 4 describes related work. In Section 5, we describe the five iterations and the model we developed . Examples demonstrating the usefulness of the model is also given in this section. Section 6 concludes.

2. Theoretical foundations

2.1. Gestalt principles
As light passes through the eye, it forms an image on the retina at the back of the eye (Young, Freedman, & Ford, 2007). We see the image, but what we perceive is something else (Shepard & Levitin, 2002). Instead of perceiving individual and distinct objects in the world, we perceive objects that are logically connected to each other (Kimchi, Behrmann, & Olson, 2003). The Gestalt school of thought puts it that our perception of the world is influenced by the way we group and segregate the stimuli (Kimchi et al., 2003). Perceptual organisation is a neuro-cognitive process that dictates how we perceive objects in physical space and how we interpret some objects as being distinct from others and yet other objects as part of a whole (Helm, 2014). The Gestalt School of thought has identified multiple principles of perception, including grouping by proximity and the principle of good

continuation. The strength of grouping is inversely proportional to the distance between the elements (Bergman, 2009); therefore, the closer the elements are to each other the stronger our perception is that these belong to the same group. Perceptual grouping by good continuation puts it that we tend to follow a gentle curve and not deviate from this path when the curve is interrupted by sudden changes (Chandler, 2007).

2.2. Semiotic theory

Krippendorff (1989) developed a model (Figure 1) that illustrates the relationship that exists between an artefact, the designer, and the user. This model separates the designer who creates the artefact from the user who makes sense of the artefact when observed in context. Artefacts used in tangible programming environments are created this way. For the user to find meaning in the artefact requires an initial mental exercise that must be repeated after some time has passed and the meaning associated with the form has faded from memory.

Our approach differs; instead of separating the designer from the user, we propose that the two roles be incorporated into a single person called the designer-cum-user. Using this approach, the user creates a personally meaningful artefact that represents the target product. Although not confirmed, we anticipate that this approach reduces the initial cognitive load on the user when he uses the artefact as a program element. Figure 2 is Krippendorff's model adapted to reflect our approach. This model does not include the reason the artefact is created. In our model (Figure 4), we make explicit the reason the artefact is created and indicate it as the instantiation of the system architect's algorithm (labelled B9). Algorithms include concepts such as turning off a light or to draw a line on the computer display.

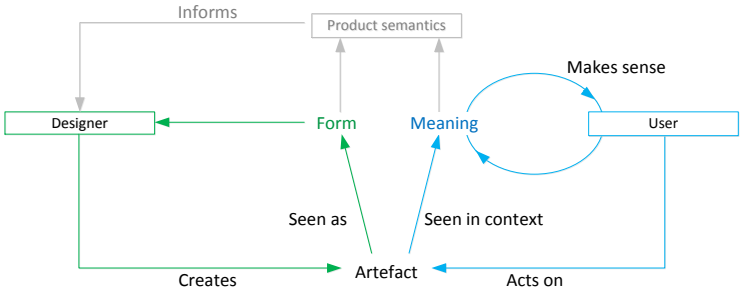


Figure 1- Krippendorff's model of the user and the designer's view of an artefact.

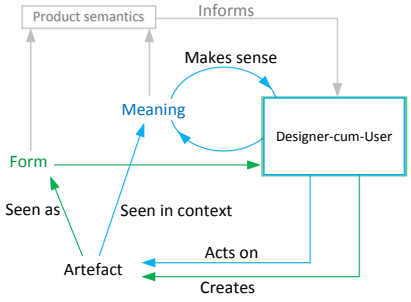


Figure 2 - Krippendorff's adapted model reflects the case where an individual is both the designer and the user.

2.3. Combining Semiotic theory and Gestalt principles in a programming environment

Our T-Logo (Andrew Cyrus Smith, 2014) programming environment explicitly incorporates the Gestalt principle of grouping by proximity to associate objects with each other and result in a program element. The arrangement of objects within a group is inconsequential. This is similar to how two English language sentences convey the same meaning: Consider the meaning of the sentences "the red car" and "the car is red". Both contain a noun and an adjective that describes the noun. Our approach is that when the words "car" and "red" are in close proximity to each other, then they are considered to be related to each other. Figure 3 illustrates this concept. Written in textual form and using an object oriented language (Java is an example), this can be expressed as `car.colour = red` where "car" is a programmatic object.

In terms of the Saussurian (Saussure, 2011) model, the two objects on the left in Figure 3 are the signifiers and the item on the right is the signified. In terms of Peirce's (Peirce, 1935) semiotic model, the two objects on the left are representamen and the imaginary object on the right is the interpretant. It is the process of semiosis that binds the representamen to the interpretant and it is the

representative character of the toy car and the presentative character of the red cloth that, when combined, result in the red car on the right. In terms of Peirce's model, the cloth is a qualisign and the car is an icon.

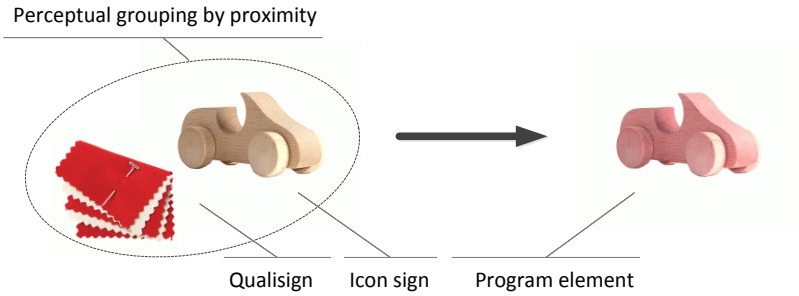


Figure 3 - An adjective describes the noun.

Semiotics research considers the meaning that objects hold for individuals and recognises the fact that the meaning one person attaches to an object may be different to the meaning another person attached to the same object. Our model of a tangible programming environment also makes provision for this difference in perception by using two representations of the same concept: One representation is in the form of a user-chosen tangible object and the system software developer chooses the second representation (being an identification number in the form of an optical marker). The two representations are reconciled when the user attaches the marker to the selected object.

3. Methodology

We embarked on our research without knowing in advance what theory would support our work. Our departure point was the domain of computer programming and as we discovered later, the general research domain of psychology would ultimately provide the theoretical underpinnings for our work. These are the theory of Semiotics and the principles of Gestalt. We therefore had a starting point but did not know what was missing in our body of knowledge.

Subsequently, we identified the Design Science Research Methodology (Hevner, March, Park, & Ram, 2004) as suitable to guide our research. This methodology recognises that at times a researcher has to rely on experience, intuition, and trail-and-error (Hevner et al., 2004) (and so did we). Vaishnavi and Kuechler (2015) put it that no single knowledge base is complete and this resonates well with our experience in this project when we had to access knowledge beyond the research domain of Computer Science by accessing Psychology domain knowledge. We applied Vaishnavi and Kuechler's (2008) process model since it makes explicit that the knowledge base is incomplete when research commences.

4. Related work

Although a number of tangible environments apply the Gestalt principle of proximity, they do so without making this theoretical foundation explicit. None of these programming environments support the user in using personally meaningful objects as program element representations. Our research identified a group of environments that implicitly use Gestalt principles: The principle of grouping by proximity is evident in ReacTable (Jorda, Kaltenbrunner, Geiger, & Bencina, 2005) while the Aggregate Cube in Blackwell and Hague's (2001) Media Cubes relies on grouping by common region.

The Gestalt principle of good continuation can be seen in programming environments such as Tern (Horn & Jacob, 2007) and GameBlocks (Andrew Cyrus Smith, 2007). Perlman's (1974, 1976) TORTIS slot machine is the first documented tangible programming environment and it enforces the Gestalt principle of good continuity. Tern and GameBlocks both infer good continuity along straight

lines; however, Gallardo et al.'s (2008) Turtan takes this a step further by including curved trajectories in the program layout.

4.1. Navigation Blocks

Camarata (2002) developed a database query system based on tangible cubes that each represent a thing, time, place, and a person in a database. Each of the six sides represents a unique place or person. By placing a cube next to a second cube, a database query is constructed that can be written as a logical AND condition. Camarata suggested, but did not implement, OR and NOT logical conditions because the author was concerned that the additional expressions would result in an interface that is less understandable. The awkward cube combination suggested to represent logical OR is easily avoided by applying the Gestalt principle of grouping by proximity: Assuming items grouped closely together implies that everything should hold true in that grouping, then the grouping constitutes a logical AND. Conversely, if objects are in separate groups, then each group is considered independently of the others and are therefore a logical OR representation.

4.2. Media Cubes

Media Cubes inspired our approach to use grouping by proximity in our solution. Blackwell and Hague's (2001) ontological programming paradigm includes sensing of events and actuation to change something in the environment. Events in a domestic setting include the ringing of the doorbell and the sounding of an alarm clock. Actuation includes changing a television channel or initiating a video recording. A Media Cube represents an "abstraction" and includes the change of state of a device. Media Cubes are combined to form small programs and rely on close proximity to other cubes to form a program. Although not implemented, Blackwell (2001) suggests that an object that represents time might actually look like a clock. This fits well with our overall aim and our model makes provision for the user to select his own time representamen. Association is achieved by placing the cube next to the appliance being controlled (A.F. Blackwell & Hague, 2001).

5. T-Logo

5.1. The initial iterations

The first two iterations explored placing cubes in a linear sequence. The tangible programming environments that resulted from these iterations are respectively called GameBlocks I and II. The tangible elements in GameBlocks I (Andrew C. Smith, 2006) are constructed using acrylic sheets whereas those in GameBlocks II (Andrew Cyrus Smith, 2008) make use of soft closed-cell foam. The third iteration considered hand-made programming elements carved from natural stone as an attempt to get the user closely involved in the design of the program artefacts. The resultant implementation is called RockBlocks. The fourth design explored repurposing everyday materials as program objects and resulted in a tangible programming system called Dialando (Andrew Cyrus Smith, 2010). Learning that emerged from the first iteration prompted the application of Gestalt principle of good continuation in the three iterations that followed. The Gestalt principle of grouping by proximity was introduced in the fifth and final iteration.

5.2. The final iteration

The final iteration incorporates both Gestalt principles of grouping by proximity and the principle of good continuation. It also includes Semiotic theory by making the user responsible for choosing the objects that represent program elements. Program elements are tangible objects that play a specific role in the program and elements include parameters, actions, and states. An object becomes an element when the user associates the object with a role. Association is done using a numbered optical marker (also known as a fiducial) attached to the object. An element has two representations: The first representation is in the physical world and the second is in the digital domain. The user interacts with the element using the physical representation and the software interacts with the digital representation. This final instantiation is called T-Logo (Andrew Cyrus Smith, 2014).

5.3. The model

From our learning based on the five iterations, we designed a model that captures the constructs involved in tangible programming. The model explicitly includes elements of semiotics and Gestalt principles. Figure 4 illustrates our model of a tangible programming environment in which the user chooses personally meaningful objects as program elements.

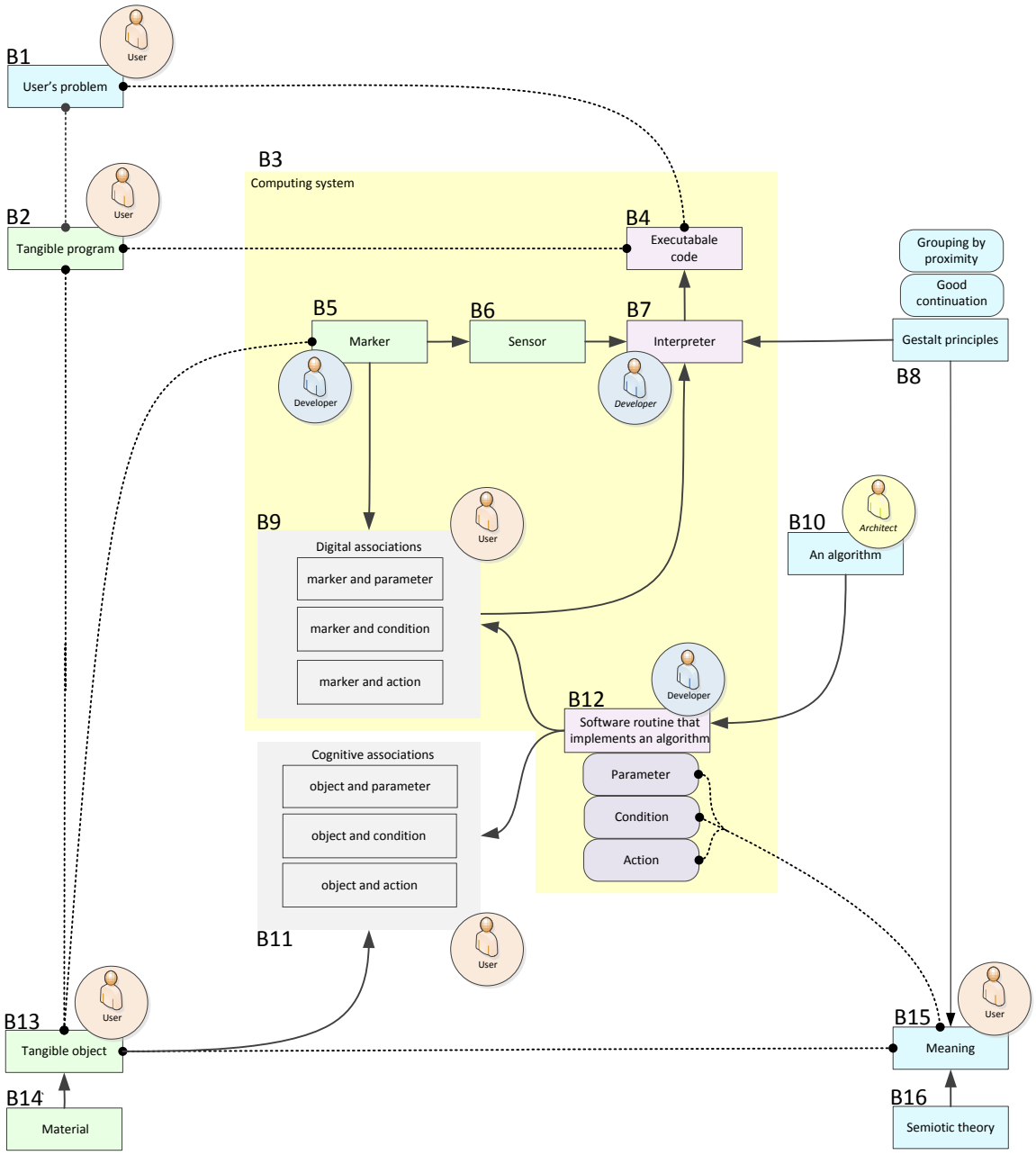


Figure 4 - A model of the T-Logo programming environment.

The model includes constructs and shows the relationships between the constructs. Constructs include persons such as the user who wants to solve a problem (B1), a system architect who specifies which algorithms (B10) to be include in the programming environment, and a software developer who interprets the algorithms and implements them as actions, conditions, and parameters (B12).

Additional constructs are a tangible program (B2) that addresses the user's problem, the computing system (B3), materials (B14), tangible objects (B13), Gestalt principles (B8), Semiotic theory (B16), meaning that objects hold to the user (B15), and the user's cognitive associations (B11) between tangible objects and software routines. The computing system in turn consists of executable code (B4) that address the user's problem, markers (B5), a marker sensor (B6), an interpreter (B7), software routines (B12), and digital associations between markers and software routines (B9), .

A developer implements software routines based on algorithms defined by the system architect. Making a drawing on paper is an example of an algorithm. An algorithm is realised using software routines that consist of actions, conditions, and parameters. Conditions reflect the world state while actions change the world state. Condition examples include facts such as "it is windy", and starting a drawing and ending the drawing are examples of actions. Parameters refine the conditions of interest, and detail how actions are applied to the world. Parameter examples include ink colour and wind speed. To the user, the routines hold meaning.

Based on Gestalt principles and Semiotic theory, the user either creates a tangible object using available materials or chooses an existing object to instantiate a personally meaningful representation of actions, conditions, or parameters. This establishes an association in the user's mind between the object and an action, condition, or parameter.

The user attaches a marker to the object and instructs the computing system to associate the marker with a particular routine. Because the marker and the object are now cognitively and digitally combined, the user and the computing system associate the combination with the same routine.

The user addresses his problem by constructing a program that incorporates personally meaningful objects representing routines. Individually, objects hold meaning to the user, while an arrangement of objects carry another meaning. Whereas the user views a program as an arrangement of one or more objects, the system interprets the program as a collection of one or more markers. Using information from the sensor, the interpreter identifies the markers and how these are arranged. It segments the arrangement according to the Gestalt principles of 1) good continuation and 2) grouping by proximity. It then produces executable code according to the user-created digital associations. The outcome is executable code that addresses the user's problem.

5.4. The model applied

In practice, a user is presented with concepts and he can craft or repurpose existing objects to signify them. For example, a hypothetical user has chosen to represent the program termination instruction using a soft toy of a sleeping puppy. In addition, three posable mannequins represent the commands FORWARD, RIGHT, and LEFT respectively. A dog modelled from clay represents GROWL (a growling sound) and a bobblehead toy giraffe is the user's interpretation of SHAKE (a shaking motion). Finally, a handheld torch represents the IR-BEAM test condition (is the beam active?) while two toy cars do the same for BUMP (has a bump been detected?). Table 1 summarises the user's selection of tangible objects for these commands and conditions. Another user may have chosen different representations.

Table 1 – User's mapping between program concepts and tangible objects, and Tern sign equivalents.

Concept	Object of user's choice	Tern sign
(Command) FORWARD		

(Command) LEFT		
(Command) RIGHT		
(Command) GROWL		
(Command) SHAKE		
(Command) Terminate the program		
(Test condition) IR-BEAM = FALSE		
(Test condition) IR-BEAM = TRUE		
(Test condition) BUMP = FALSE		
(Test condition) BUMP = TRUE		

To test our approach to having the user select tangible program objects, we chose three previously published Tern programs and show how they can be implemented using our hypothetical user's objects. The first (Figure 5) is a sequence of six program statements: FORWARD, GROWL, RIGHT, FORWARD, SHAKE, and LEFT. Since the T-Logo programming environment continually executes the user's code, we added a program termination object (the soft toy in this example) at the end of the

T-Logo sequence in Figure 5b. The dashed line highlights the principle of good continuation. Six slanted lines show the correlation between the Tern (Figure 5a) and the T-Logo implementations.

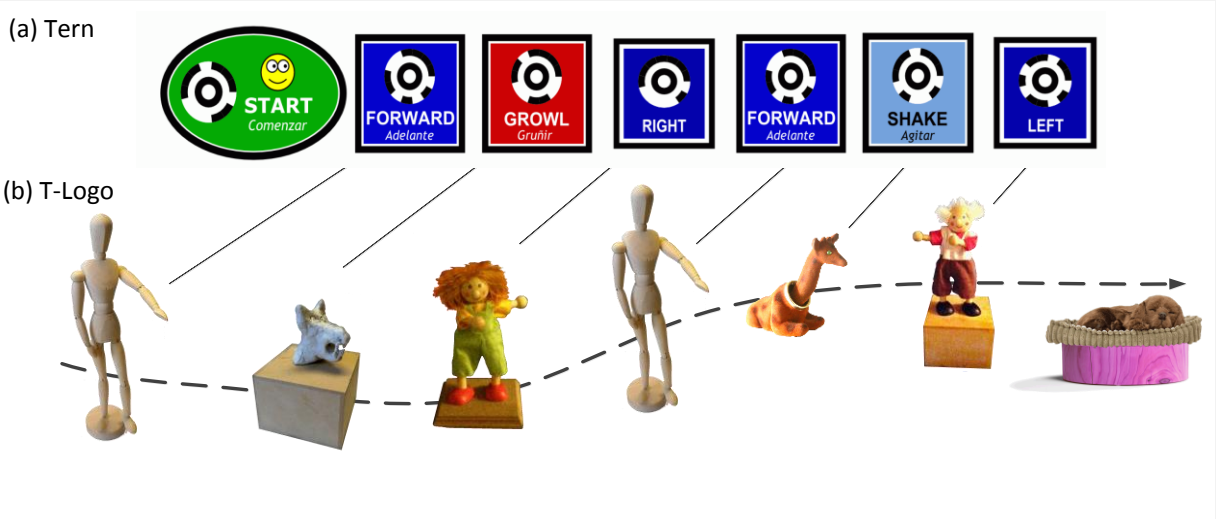
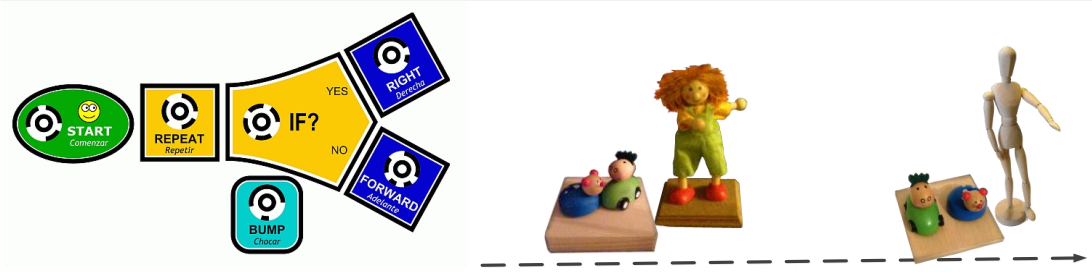


Figure 5 - Example of a linear program.

The second example (Figure 6) demonstrates a conditional IF statement and the use of a parameter. This is a typical code segment for a toy robot and determines the robot movement when a bumper (being the parameter) is activated. Figure 6a is a program using the Tern language while Figure 6b is the same program logic implemented using T-Logo.



(a) Tern (b) T-logo

Figure 6 - Example of a conditional statement in a loop.

The third example illustrates a logical AND expression. In this example, the compound AND expression in Figure 7a has been rewritten as four expressions and implemented using T-Logo. The original expression is:

```
IF (IR-BEAM = TRUE) AND (BUMP = TRUE) THEN do RIGHT ELSE do FORWARD.
```

Figure 7b shows the Tern implementation. Equivalent logic can be composed and the result is expressed in Figure 7c using the T-Logo language:

```
IF (NOT-IR-BEAM = TRUE) AND (NOT-BUMP = TRUE) THEN do FORWARD
IF (IR-BEAM = TRUE) AND (NOT-BUMP = TRUE) THEN do FORWARD
IF (NOT-IR-BEAM = TRUE) AND (BUMP = TRUE) THEN do FORWARD
```

```
IF (IR-BEAM = TRUE) AND (BUMP = TRUE) THEN do RIGHT
```

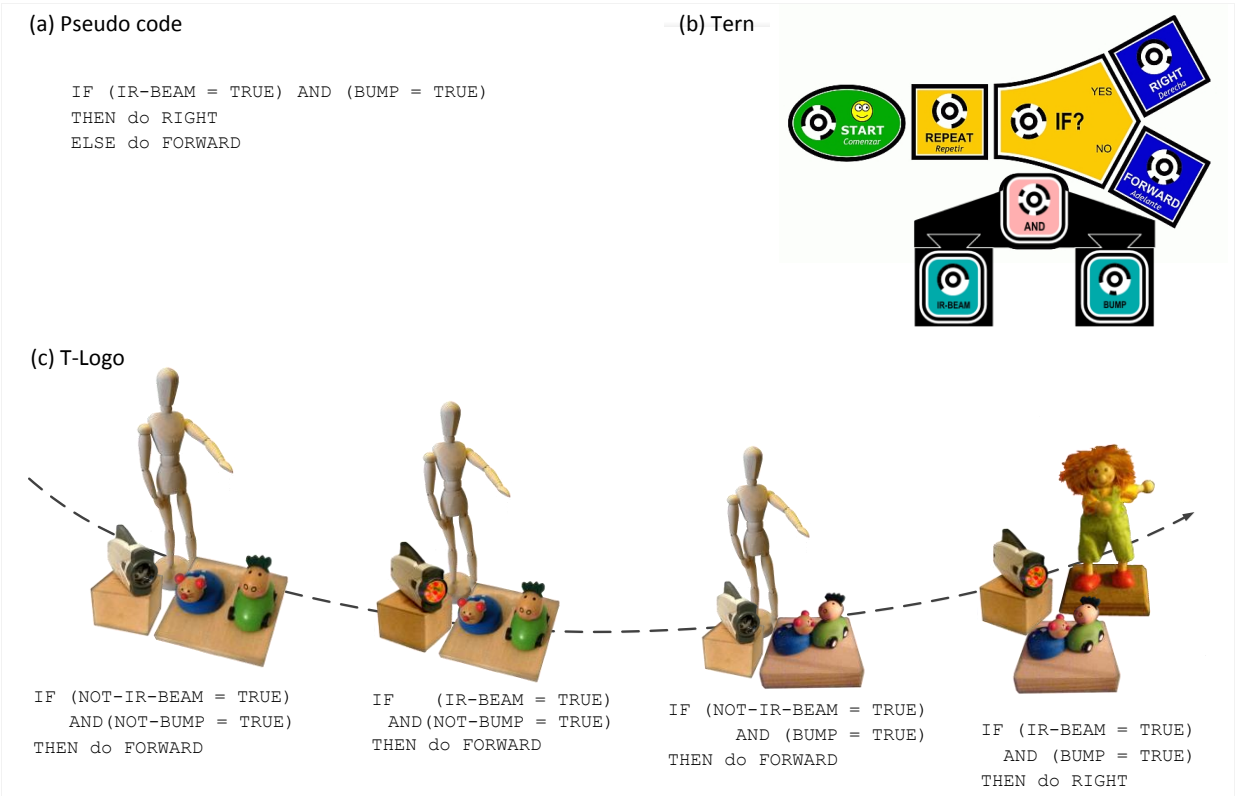


Figure 7 - Example of the logic AND conditional statement.

6. Conclusion

We have set out to develop a tangible programming environment in which the user can choose his own objects to represent program elements but discovered that we had insufficient knowledge on how to proceed. Our search for an appropriate research methodology identified Design Science Research as appropriate. We subsequently found theory and principles in the Psychology domain to underpin our research, specifically Semiotic theory and Gestalt principles. An analysis of existing tangible programming environments revealed that some Gestalt principles were implicitly present.

We presented a model for a tangible programming environment in which the user can use personally meaningful objects and apply Gestalt principles when constructing a program. Finally, to test the effectiveness of our T-Logo programming environment we constructed three hypothetical programs that are equivalent to previously published Tern code segments.

7. References

Bergman, M. (2009). *Peirce's philosophy of communication: The rhetorical underpinnings of the theory of signs*. Continuum studies in American philosophy. London: Continuum International Publishing Group.

Blackwell, A. F., & Hague, R. (2001). AutoHAN: An architecture for programming the home. *Human-centric computing languages and environments, 2001. Proceedings IEEE symposia on* (pp. 150–157). IEEE.

Blackwell, A. F., & Hague, R. (2001). Designing a programming language for home automation. *Proceedings of the 13th annual workshop of the Psychology of Programming Interest Group (PPIG 2001)*, 85–103.

Camarata, K., Do, E. Y.-L., Johnson, B. R., & Gross, M. D. (2002). Navigational blocks: Navigating information space with tangible media. *IUI '02: Proceedings of Intelligent user interfaces*

(pp. 31–38). San Francisco, California, USA: ACM Press. doi:http://doi.acm.org/10.1145/502716.502725

Chandler, D. (2007). *Semiotics- the basics* (second.). Abingdon, Oxon: Routledge.

Eckel, B. (2006). *Thinking in Java* (fourth.). Boston: Prentice Hall Professional.

Gallardo, D., Julia, C. F., & Jorda, S. (2008). TurTan: A tangible programming language for creative exploration. *TABLETOP 2008. 3rd IEEE International workshop on horizontal interactive human computer systems* (Vol. 10, pp. 89–92). doi:10.1109/TABLETOP.2008.4660189

Helm, P. A. van der. (2014). Oxford handbook of perceptual organization. In J. Wagemans (Ed.), . Oxford, United Kingdom: Oxford University Press.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105. Retrieved from http://www.jstor.org/stable/25148625

Horn, M. S., & Jacob, R. J. K. (2007). Tangible programming in the classroom with Tern. *CHI '07 extended abstracts on human factors in computing systems* (pp. 1965–1970). San Jose, CA, USA: ACM Press. doi:http://doi.acm.org/10.1145/1240866.1240933

Jorda, S., Kaltenbrunner, M., Geiger, G., & Bencina, R. (2005). The reacTable. *Proceedings of the International computer music conference (ICMC 2005)*. Barcelona, Spain.

Kimchi, R., Behrmann, M., & Olson, C. R. (Eds.). (2003). *Perceptual organization in vision–behavioral and neural perspectives*. Lawrence Erlbaum Associates, Inc.

Krippendorff, K. (1989). On the essential contexts of artifacts or on the proposition that "design is making sense (of things). *Design Issues*, *5*(2), 9–39. Retrieved from http://0-www.jstor.org.oasis.unisa.ac.za/stable/1511512

Peirce, C. S. (1935). *The collected papers of Charles Sanders Peirce*. (C. Hartshorne, P. Weiss, & A. W. Burks, Eds.) (Vol. 1–8). Cambridge: Harvard University Press.

Perlman, R. (1974). *TORTIS: Toddler's own recursive turtle interpreter system*. MIT Artificial Intelligence Lab.

Perlman, R. (1976). *Using computer technology to provide a creative learning environment for preschool children.* (No. 24 (Logo Memo), MIT Artificial Intelligence Lab Memo 360). MIT Artificial Intelligence Lab.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., et al. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67. doi:http://doi.acm.org/10.1145/1592761.1592779

Saussure, F. de. (2011). *Course in general linguistics*. (P. Meisel & H. Saussy, Eds.). New York: Columbia University Press.

Shepard, R. N., & Levitin, D. J. (2002). Foundations of cognitive psychology - core readings. In D. J. Levitin (Ed.), (pp. 503–514). The MIT Press.

Smith, A. C. (2006). Tangible cubes as programming objects. *Proceedings of the 16th International conference on artificial reality and telexistence–workshops (ICAT'06)* (pp. 157–161). Hangzhou, China: IEEE Conference Publications. doi:10.1109/ICAT.2006.121

Smith, A. C. (2007). GameBlocks: an entry point to ICT for pre-school children. *Meraka Innovate Conference*. CSIR Convention Centre, Pretoria South Africa. Retrieved from http://hdl.handle.net/10204/1778

Smith, A. C. (2008). A low-cost, low-energy tangible programming system for computer illiterates in developing regions. *4th International workshop on technology for innovation and education in developing countries (TEDC)*. Retrieved from http://playpen.meraka.csir.co.za/ acdc/education/TEDC 2008 Proceedings - Technology for Innovation and Education in Developing Countries: 978-0-620-43087-6/Smith_08.pdf

Smith, A. C. (2010). Dialando: Tangible programming for the novice with Scratch, Processing and Arduino. *6th International workshop on technology for innovation and education in developing countries (TEDC)*. Retrieved from http://hdl.handle.net/10204/4048

Smith, A. C. (2014). Cluster-based tangible programming. *Digital information and communication technology and it's applications (DICTAP), Fourth international conference on* (pp. 405–410). IEEE. doi:10.1109/DICTAP.2014.6821720

Vaishnavi, V. K., & Kuechler, W. (2008). *Design science research methods and patterns: innovating information and communication technology*. Auerbach Publications.

Vaishnavi, V. K., & Kuechler, W. (2015). *Design science research methods and patterns: innovating information and communication technology*. CRC Press.
Young, H. D., Freedman, R. A., & Ford, L. (2007). *University Physics*. Addison Wesley.