The Pwn4g3 Kube

A Tangible Interfacing Device

Adam S. Fox
Jonathan S. Fulton
Nathaniel E. Huebscher
Brendan C. Lyons

This project represents our own work, in accordance with university regulations.

1 Introduction

Tangible user interfaces (TUIs) provide the means to interact with computers in a physical manner. The Pwn4g3 Kube is an example of a TUI that exploits the intuitive properties of a three dimensional object. By giving the user the freedom to interact with a physical cube, we believe that the user's experience in using certain applications will be enriched. The Pwn4g3 Kube affords natural gestures, such as flipping a cube's face or rotating the cube about an axis. To provide the user with a unique and entertaining experience, we map these gestures to various software applications. This paper presents the Pwn4g3 Kube and three applications: *kubeLAUNCH*, *kubeMAZE*, and *kubeSECURITY*.

kubeLAUNCH is an application that allows the user to easily open up installed computer programs. We felt that the Windows "Quick Launch" system and the Mac "Dock" menu were both inadequate, because of their physical location on the screen. The Pwn4ge Kube provides the user with a tangible selection process; one must simply flip the cube's face. By mapping gestures to the *.exe* file of an application, we provide the user the ability to easily open new instances of his/her favorite software.

kubeMAZE is a simple maze game where the user attempts to escape by rotating the Pwn4g3 Kube. By interpreting the user's input gestures as on-screen movement direction, one can gain an enriched physical "feel" for a simple puzzle. kubeMAZE is designed with younger children in mind; a child should easily learn to use the Kube like a toy.

kubeSECURITY is an application that is built on the idea that passwords are remembered as physical gestures. As such, it might be possible to remember a sequence of Kube flips as a password, especially for those who are disabled and cannot use a keyboard properly. This application implements that idea, where the user can play around with using the Kube to create a password.

2 Project Break-up

- Adam: Physical Cube design. Initial Bluetooth setup. Implementation of Java Serial Communication API. Aid with initial data analysis and face recognition.
- Brendan: Aid with Initial Bluetooth setup. Data parsing. Initial data analysis and face recognition.
- Jonathan: Software design.
- Nate: Gesture recognition. Testing the finished product and experiment design.

3 Design

3.1 Hardware, Connectivity, and Initial Data Processing

3.1.1 Materials

- WiTilt v3.0 three-axis accelerometer from SparkFun Electronics
- Clear plastic baseball display. (3 inch cube)
- Jabra A320s Bluetooth Stereo USB Adapter
- Poster wall mounts

3.1.2 Setup

The WiTilt v3.0 was a very useful three-axis accelerometer that allowed for Bluetooth communication. When properly connected, the device outputs X, Y, and Z acceleration data along with gyroscope data R. The WiTilt fits perfectly inside of the three inch clear plastic baseball display cube. We used two adhesive poster wall mounts to affix the WiTilt in the center of the cube, trying to make it as nearly perpendicular or parallel to the sides of the cube as possible. Since the display case is very simply to open and close, this setup allowed for easy access to the device for turning it on and off and for charging. In addition, by making the device as in line with the sides as possible we were able to ensure uniform output from the accelerometers.

We then proceeded to sync the device with our computer. In order to make this as portable as possible between our laptops, we purchased the Jabra Bluetooth adapter, which allowed us to create Bluetooth serial ports via our USB ports. Next, we installed a very useful set of Java libraries, GiovynetSerialPort (http://www.giovynet.com/serialport.html), which contains basic APIs that allowed us to communicate with serial ports. After pairing the WiTilt and the computer with the built-in Bluetooth software (or the software that came with the Jabra), we were able to use Java to connect with and communicate with the device.

Each time the device was connected to the computer for a particular application, it was configured to output the raw data from the X, Y, and Z accelerometers. We also chose to receive the gyroscope data. This data is transmitted to the computer as a line, character by character, for each reading. Each line is then parsed into integers, one for each direction plus the gyroscope information. A little experimentation showed that each side of the cube was uniquely represented by these three integers. The integers were limited to between 0 and 1024. While it was on a side, however, each integer fell into one of three ranges: low (190-320), mid(450-570), and high (700-830). Depending on which of X, Y, and Z fell into which of low, mid, and high, the current top side of the cube could be easily determined. If the values of X, Y, and Z did not indicate a proper side, a default value was returned. This occurred, for example, when the cube was held askew or shaken violently.

3.2 Kube API

The Kube API offers two sets of buffers that clients can access: the four data buffers (X,Y,Z,R) and the actions buffer (FORWARD, BACKWARD, LEFT, RIGHT, OVER). The data buffers store the raw data that the Kube sends and the actions buffer stores actions as they are determined. It is critical to note that we assume the user starts with the cube in a predefined orientation, namely the orientation such that the writing on the WiTilt can be read facing upwards. In order to determine when an action occurs, it is necessary to store the current face that points up. When this face changes, an action has occurred. To determine which action occurred, it is necessary to store the current state of the Kube. Assuming the Kube's orientation is only changed when it is flipped (i.e. the user does not spin the cube), then it is always possible to know the current orientation of the Kube. Every rotation (FORWARD, BACKWARD, LEFT, RIGHT, but not OVER because it actually maps to four different orientations) is a simple permutation operation determined by the previous side and the new side. Once the action is determined, the orientation is updated and the action is added to the actions buffer. In summary, the action definition function works as follows:

1. If the current side changes then
2. The action is determined by the current orientation and the new side.
3. The action is added to the actions buffer.
4. The orienation is adjusted according to the action.

This is the only major algorithm we use for the Kube API other than the face-determination algorithm previously described. It would be possible to define other actions, such as "shake" or "twisting", but we currently do not have these implemented.

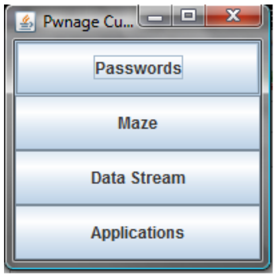


Figure 1: rootMenu.

3.3 User Interface

We spent quite some time developing what we considered to be a usable and practical interface for testing the Kube. It was developed entirely in Java. The root menu, shown in Fig

(1), provides a simple interface for accessing the available applications for the Kube. This interface consists of four buttons linked to the four cube applications: Passwords, Maze, Data Stream, and Applications.

Initially, we developed an application that graphically displays the data provided by the Kube. It includes four plots, one each for the X, Y, Z and R (gyroscope) data streams. The data is plotted in real time and a screenshot of this application is shown in Fig. (5). This application was quite useful to explore the capabilities of the Kube and its sensitivity and responsiveness to user input.

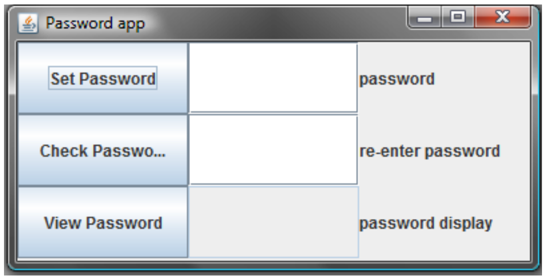
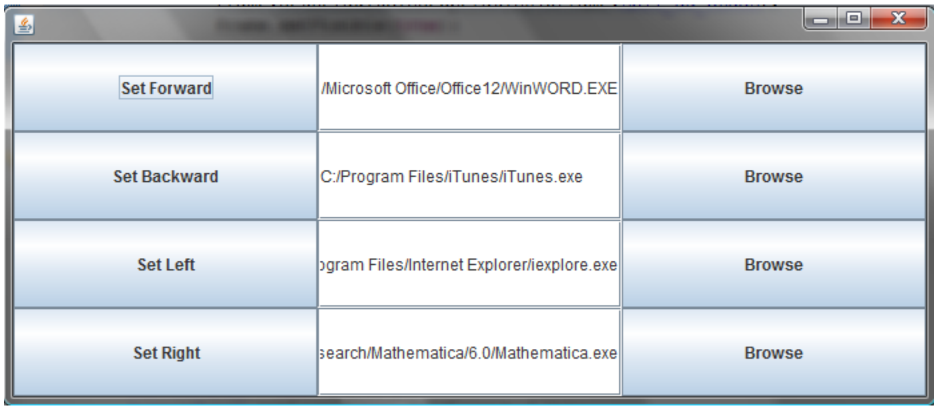


Figure 2: kubeSECURITY menu

Next, we developed an interface for the kubeSECURITY application. This interface, shown in Fig (2), consists of three buttons and two entry fields. Passwords can be entered directly using either the Kube or the keyboard. In order to establish a password, the user must enter the desired password into the two fields, and then click on Set Password. Then, the user can enter sequences of gestures into the password field and check it against the password using the Check Password button. Finally, if the user desires to view the password that he/she created, the View Password button may be used.



Set Forward	/Microsoft Office/Office12/WinWORD.EXE	Browse
Set Backward	C:/Program Files/iTunes/iTunes.exe	Browse
Set Left	ogram Files/Internet Explorer/iexplore.exe	Browse
Set Right	search/Mathematica/6.0/Mathematica.exe	Browse

Figure 3: kubeLAUNCH Menu

To exploit the full capabilities of kubeLAUNCH, we developed an interface, shown in Fig (3), which can be used to select new programs to map to the actions of forward, backward, left and right. First, the user can click on the Browse button, which allows him/her to select a new executable program to map to the desired action. After clicking the corresponding Set button, the change is complete and the cube is ready to launch the desired programs whenever desired.

Lastly, we developed a user interface for the kubeMAZE application, shown in Fig. (6). The purpose of this interface is to test the usefulness of the cube as on-screen navigation device. The application is quite robust, and can be used to generate a maze of an N by N grid. As can be seen from the figure, the current position is indicated by a small red circle. The user can navigate the maze using input from the cube, the arrow keys, or the buttons on the left of the screen. When the maze is completed, the maze restarts from the beginning. If the user desires a new maze, he/she may click the New Maze button with either a new size or with the same one. If the user wants to see a solution shown on the maze, he/she may click the Show Solution button, which computes the solution to the maze using a DFS algorithm and plots the solution on the maze. Lastly, the Start/Stop button on the upper left hand corner starts and stops connectivity with the cube, respectively. If the user accidentally disorients the cube, he/she may use this same button to reorient it.

4 Testing the Pwn4g3 Kube

Experimental design and implementation is integral to the success of the cube as a viable HCI device. We performed several primary experiments to test the functionality of the cube. First, we conducted an experiment to test the effectiveness of the cube with the maze application. Our goal was to compare the effectiveness of the cube as input device with that of traditional input controls: namely, on-screen buttons and arrow keys. Our second experiment tested the effectiveness of the cube as a method for password security protection by investigating individuals' ability to retain passwords in short term memory as well as its value as a method for maintaining long term security (e.g. preventing observers from easily copying security code).

To test the functionality of the cube with the maze application, we conducted an experiment comparing the performance of the cube with that of the on-screen buttons and the keyboard arrow keys. The on-screen buttons were designed in an unintuitive way, such that the user needs to pause before each input to think. The experiment included six participants, picked from a pool of interested persons in a computer cluster in Princeton University. Participants were asked to complete the maze puzzle (with solution shown) for mazes of size $N = 10, 15$,and 20. The following is the data we collected (units are in seconds):

Participant 1	$N = 10$	$N = 15$	$N = 20$	Participant 2	$N = 10$	$N = 15$	$N = 20$
Cube	20.3	39.1	86.5	Cube	18.8	53.66	75.67
Buttons	32.3	60.4	150.9	Buttons	29.68	69.41	110.22
Arrow Keys	7.7	13.45	30.1	Arrow Keys	8.17	16.39	23.3
Participant 3	$N = 10$	$N = 15$	$N = 20$	Participant 4	$N = 10$	$N = 15$	$N = 20$
Cube	15.52	18.72	89.33	Cube	34.86	63.91	46.52
Buttons	35.58	33.6	175.8	Buttons	47.94	92.47	70.47
Arrow Keys	8.7	10.71	39.65	Arrow Keys	8.81	17.51	12.74
Participant 5	$N = 10$	$N = 15$	$N = 20$	Participant 6	$N = 10$	$N = 15$	$N = 20$
Cube	15.31	25.03	86.28	Cube	24.07	55.06	61.25
Buttons	24.59	34.91	97.78	Buttons	33.95	63.76	47.91
Arrow Keys	6.49	9.93	23.38	Arrow Keys	8.06	17.27	16.26

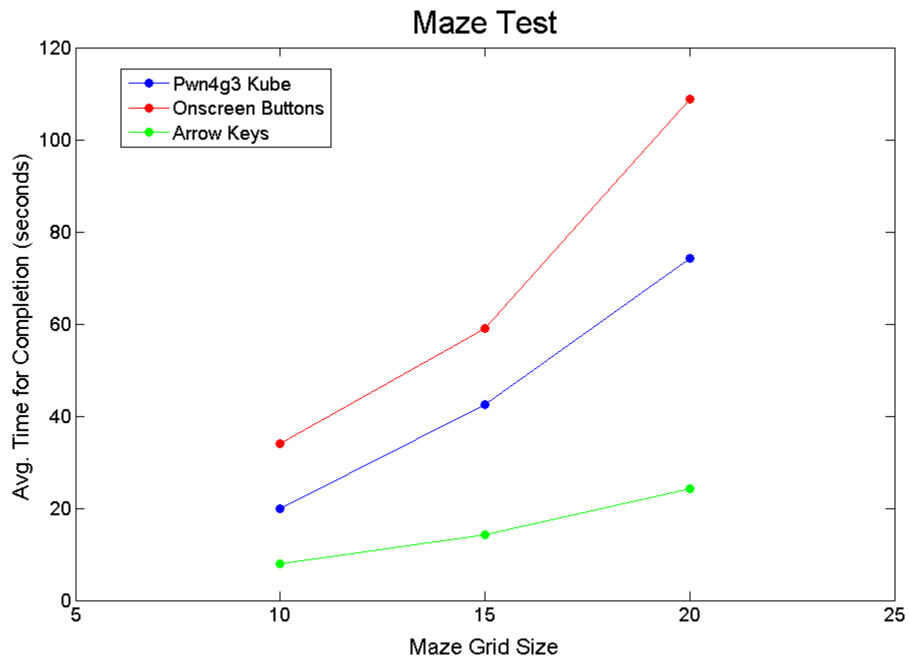


Figure 4: Charting performance on kubeMAZE

Examining this data, we can clearly see that the cube performed better than the onscreen buttons but worse than the keyboard arrows in all cases. The time taken to complete the maze increases dramatically between the three maze sizes for all input devices. Figure (4) is a plot of the average of all participants' performance for each maze size for each input device. It is shown that the time increases more dramatically for the buttons and the cube than for the arrow keys.

Comparing the performance of the three input methods, the arrow keys perform much better than either the buttons or the cube. This could be attributed to various causes. First, the arrow key interface is very intuitive and very familiar to everyone who is comfortable using computer (which all of the participants were). Second, use of the arrow keys requires no processing time; in other words, once the user recognizes what action is required, he/she can enter the correct key without having to think and the computer does not lag. Alternatively, due to the intentionally cumbersome design of the button controls, the user requires time to process every step. Also, even though the cube's actions are as intuitive as the arrow keys' there is an inherent lag in the processing time that prevents the user from operating as quickly and fluidly as he/she may otherwise be able. This is a limitation that should be explored in future study, as the difference between the performance of the cube and the arrow keys may be minimized if the user can enter actions as quickly as he/she can think of them and perform them.

Qualitatively, participants in this study responded positively to the cube. Although they felt that it was not as easy to use as the keyboard, they all stated that the cube was a much more interesting input device and enjoyed using it to complete the maze and looked forward to using the cube for future applications. In particular, the kubeLAUNCH program was met with particular critical praise.

The experiment we performed to test the cube's efficacy as a security device was quite simple. The four participants attempted to memorize randomly generated five, seven, and nine gesture long passwords and enter them correctly using the password GUI. The participants demonstrated mixed success. All four demonstrated about 80% accuracy when entering passwords with a length of five. However, the participants exhibited only about a 20% success rate with passwords of length of seven, and it was shown that gesture passwords of length nine are almost impossible to retain in short term memory.

The lackluster performance of the participants does not necessarily indicate a flaw in the design of the cube or its inability to be used as a security device. Rather, passwords are not meant to be used as short term memory objects, but quite long term. It remains to be shown what the longest gestural password someone can effectively memorize for security protection. One thing is certain, however; it would be very hard for a lurking onlooker to be able to ascertain a cube-user's password simply by looking on, and therefore, may add additional security to normal passwords. Furthermore, we have yet to test the cube as a security device for small children and disabled persons, for whom we predict the cube would serve as a very useful security device.

5 Future Work

Although our project was clearly successful, there are many features and funcionalities that we could add to the Pwn4g3 Kube in the future. One basic addition we could make is the addition of more gestures, including a “shake” and/or “twist” gesture. This would allow the user to perform various functions while using the cube for on-screen navigation, including item selection and deletion. We also would like to be able to use the gyroscope data from the Witilt to detect rotation in the cube so that, even if the user performs an illegal rotation, the cube will still remain oriented.

We would also like to implement another version of kubeLAUNCH in which the opening of applications is mapped to sides instead of gestures. We initially tested the application with this mapping, but eventually decided to use our current mapping, deeming it more intuitive for the user. However, there are advantages to using the sides of the cube, not least of which is being able to access six applications instead of four. Actions like opening additional windows or closing the application can be controlled by the additional gestures described above.

It is our belief that the Pwn4g3 Kube would be able to significantly aid people who are unable to use traditional interface devices or who wish to be acclimated to the use of computational devices through a tangible interface. The Kube would especially be useful to small children who are learning how to use a computer and to mentally and physically disabled persons who, for whatever reason, may not be able to use a traditional interface. For example, one could imagine how incredible it would be to give a person with severe Parkinson’s disease the ability to control a computer and even password protect his/her information when before the buttons and gestures of the keyboard and mouse interface were too small and too sensitive for them. Therefore, for future work, we would like to have the opportunity to work with some of these demographics in order to test the cube’s effectiveness in this application.

The cube interface has incredible potential even beyond the scope of the Pwn4g3 cube project. In thinking about where the cube idea may go in the future, one of the members of our group, Mr. Jonathan Fulton, has envisioned a device he calls Universal Kube Computing (UKC). This device, he holds, would add much functionality to the current Kube and would even be able to completely replace the role of the mouse, keyboard, and remote control for the TV. The basic design of such an interface would be a cube with a touchscreen LCD or oLED screen on each side of the cube, allowing the user manipulate items of the display of each of side. By wirelessly connecting to the a nearby PC, the user could have a small copy of his/her display appear on the top face of the Kube, which combined with the Touch Screen features would allow the Kube to act as a pointing device. Additionally, the Kube could be configured as a universal remote, potentially allowing the Kube to become the center of future home entertainment systems. If and when such a device will ever exist is only for the future to tell, but Mr. Fulton’s idea currently serves as a testament to the power of the Kube.

References

[1] Hiroshi, Ishii, and Brygg Ullmer. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. Proceedings of CHI '97, March 22-27, 1997.

[2] Lertsithichai, Surapong and Matthew Seegmiller. CUBIK: a bi-directional tangible modeling interface. Conference on Human Factors in Computing Systems, CHI 2002. April 20-25, 2002, p 756-757.

[3] Van Laerhoven, Kristof et al. Using an autonomous cube for basic navigation and input. ICMI '03: Proceedings of the 5th International Conference on Multimodal Interfaces. Vancouver, British Columbia, Canada. 2003.

[4] Zhou,ZhiYing et al. Interactive Entertainment Systems Using Tangible Cubes. http://www.ieconference.org/ie2004/proceedings/019%20zhou.pdf, 1/13/2009. 2004.

[5] Kranz, Matthias et al. A Display Cube as a Tangible User Interface. In Adjunct Proceedings of the Seventh International Conference on Ubiquitous Computing (Demo 22). 2005.

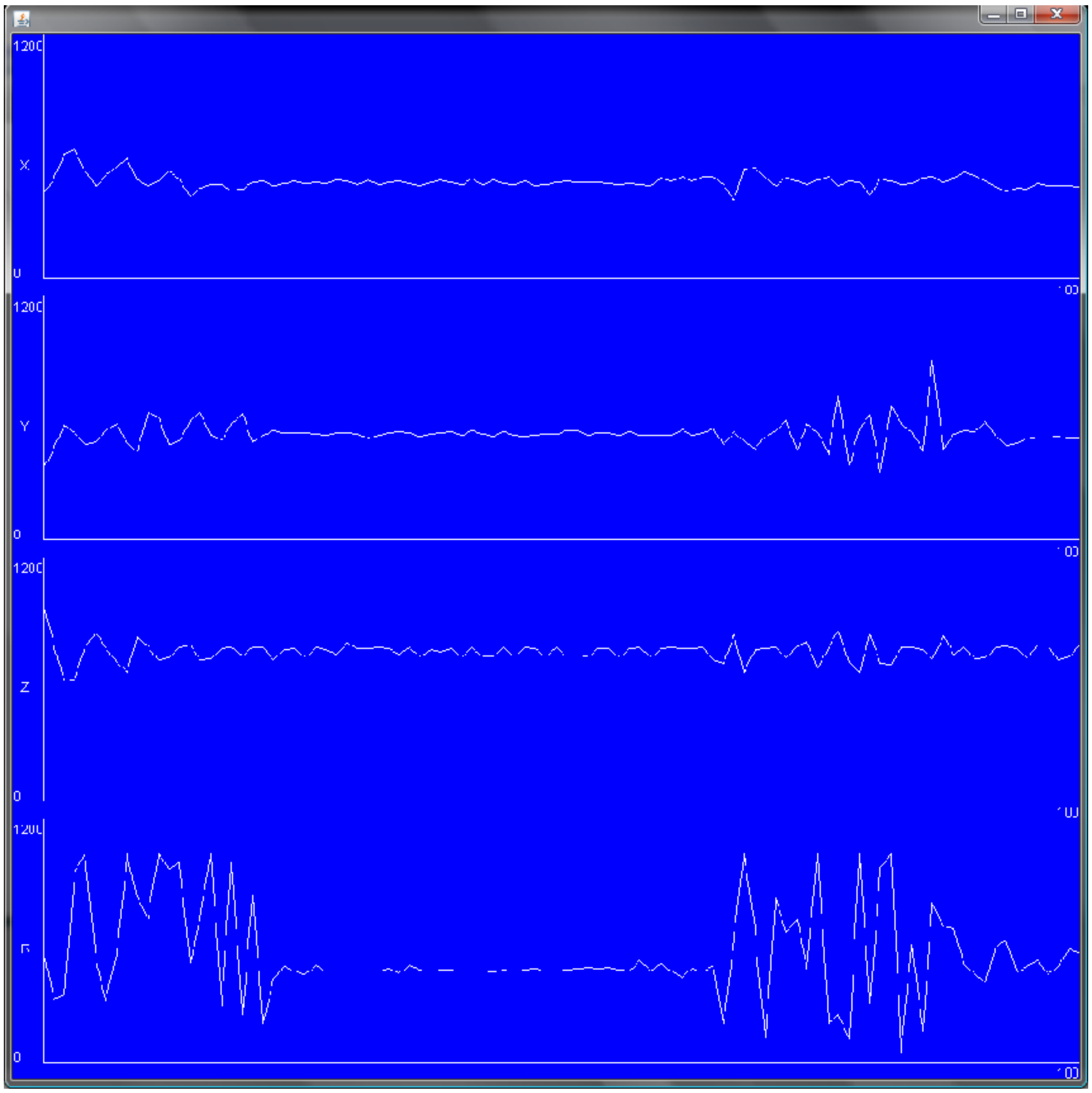


Figure 5: dataMenu. A real-time representation of X, Y, Z acceleration and gyroscope data.

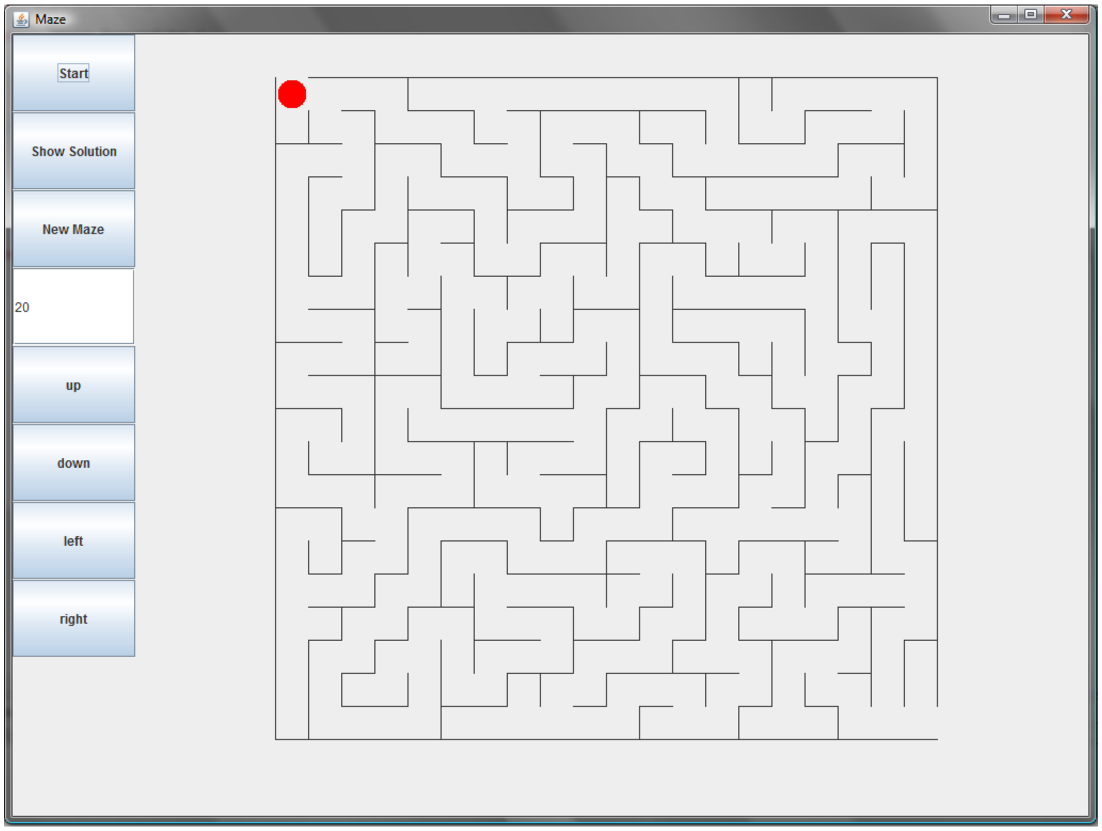


Figure 6: mazeMenu. A puzzle game that is solvable via Kube gestures.