Panic Room: Experiencing Overload and Having Fun in the Process

Björn Bankowski, Thiemo Clausen, Dirk Ehmen,
Maximilian Ernestus, Henning Hasemann, Tobias Jura,
Alexander Kröller, Dominik Krupke, and Marco Nikander

Technische Universität Braunschweig, Germany
{b.bankowski,t.clausen,d.ehmen,m.ernestus,h.hasemann,t.jura,a.kroeller,
d.krupke,m.nikander}@tu-bs.de

Abstract. We present the "Panic Room", an ambient system in the form of a game, where a player has to perform an ever-growing number of parallel tasks until he is overloaded. The game is built in a way to deduce construction and design principles for pervasive environment, as it allows for experimenting with design anti-patterns, disguised as game elements.

1 Introduction

Recent advances in embedded devices allow for a simple integration of everyday household objects with the Internet, forming the Internet of Things. In the very near future, smart objects will become ubiquitous, allowing people to build complex ambient intelligence (AmI) systems at very low cost. Such systems could be used for a number of innovative and everyday tasks, such as influencing human behavior to help overcome certain disorders [Fog02,KMRA09], help in assisted living for elderly people [Dem08,GAL]. This requires the user can handle them properly, even when stressed or overstrained. Important challenges in the design of such systems are the cognitive, emotional and action-oriented effects on the user. Many people already feel helpless and stressed when working with traditional computing devices. AmI systems could increase this, with the user feeling surrounded and controlled by an automatic system. Knowing the associations between different types of design and the users emotional, cognitive and behavioral responses, is thus essential for designing systems that are accepted by the user.

The Panic Room is used to study this process, by turning this goal on its head. It presents itself as a game for one or more players; see Fig. 1. The players interact with smart objects by performing actions and gestures. According to the background story, this is necessary to prevent a submarine from sinking. The game cannot be won however; the submarine always sinks. The objective is to design the room and the interactions in a way which keep the players engaged while they are asked to perform more and more tasks, of increasing complexity, until they are finally overburdened and fail.

N. Streitz and P. Markopoulos (Eds.): DAPI 2014, LNCS 8530, pp. 241–252, 2014.

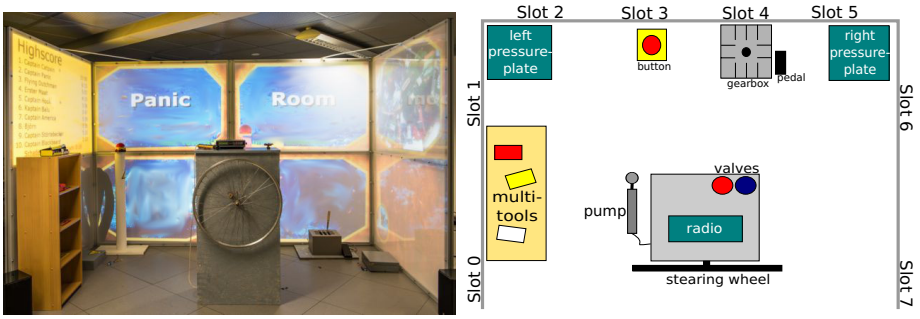

Fig. 1. The Panic Room. Left: View of an entering player. Right: Top-view schematic of object placement.

2 Design

The central goal in the design is to keep the players engaged in what they experience as fun, but at the same time to overload them with an overwhelming number of tasks. To ensure a positive experience however, avoiding frustration or a feeling of powerlessness is absolutely essential. Observing and interviewing players allows drawing conclusions about the influence and rank of different aspects of AmI systems.

In addition to observing players during the game, we interviewed players after the game and tailored the experience to balance the complexity of the game. Our observations allow linking of design choices to reactions, and the deduction of design and construction principles for usable ambient systems.

2.1 Environment

The room consists of three semi-opaque walls, covering a space of about 3×1.5 meters. Back-projection is used to display a submarine interior and an underwater environment on the walls, the overall visual impression was tailored to resemble a playful submarine impression and supported by an acoustic background with watery effects and typical submarine "pings".

Interaction with the system happens through smart objects which were placed at different locations all over the room, in a way that will require the players to move through the room in order to reach different objects. This effect is supported by the console in the center of the room. It constitutes an obstacle that must be walked around when trying to reach other objects.

The projection surface is large enough so it cannot be observed completely by a player interacting with one of the smart objects, which forces players to turn their heads in order to notice all upcoming tasks.

2.2 Smart Objects

All smart objects in the room are battery powered and operate wirelessly. They are robust enough so that a player can interact with them freely without having to be worried about breaking things. In addition to the movable smart objects, the environment is fitted with a number of RFID tags which are hidden at different places to allow device localization. While most objects allow the user to give input to the system by pushing buttons or levers, turning wheels, and even dispositioning objects, some of the objects also provide additional ways of feedback from the system to the player. This is particularly useful when it comes to overloading a player, because it makes it possible to demand the player's attention at different places and/or objects at the same time.

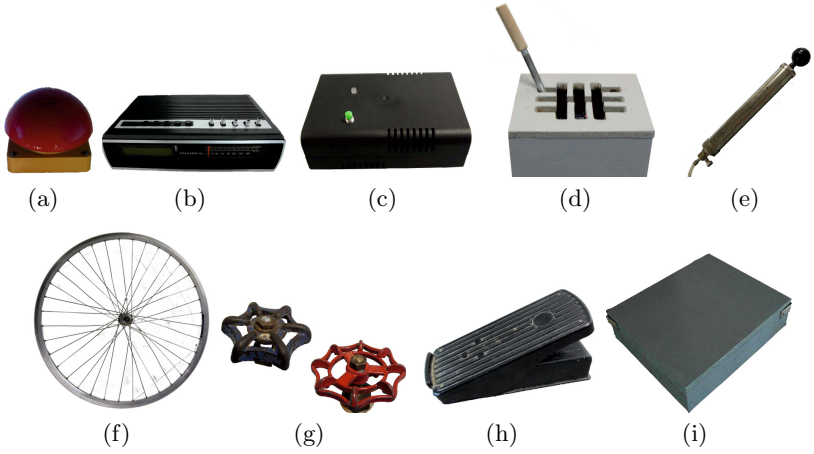

Fig. 2. Smart objects used in the Panic Room: Pushbutton (a), Radio (b), Multitool (c), Gearbox (d), Air Pump (e), Steering Wheel (f), Valves (g), Foot Pedal (h) and Pressure Plate (i)

In total there are twelve objects to interact with, chosen from nine different types:

Push-Button. The push-button object, shown in Fig. 2(a) consists of a large emergency-stop button mounted on a pillar at waist level. Due to its simplicity it is mainly used in introductory tasks and tasks which consist of interactions with multiple objects.

Radio. The radio, as seen in Fig. 2(b), is one of the more complex objects in the room. It is equipped with a card reader, five toggle switches and a rotary knob. The card reader can be used to recognize colored ID cards. It furthermore features a small 2-line text display and a beeper. Altogether the radio allows complex interactions and thus has the potential to draw the user's whole attention to this single object. It is placed on the steering wheel

console and is freely movable (although no tasks require it to be moved). Surveying the users revealed that the radio was considered the most "interesting" object, perhaps a consequence of it being the only smart object in the guise of an every-day object.

Multitools. We provide three Multitool objects like the one depicted in Fig. 2(c). These are hand-sized boxes in different colors, especially suitable for moving them around in the whole room. The boxes can detect when they are placed at specific locations within the room. Additionally, each box can determine its orientation. The boxes are equipped with a small push button and a multicolor LED to provide feedback to the player.

Gearbox. The gearbox, shown in Fig. 2(d) is a wooden box with a lever which can assume twelve different positions, mimicking an over-complicated version of the shift-stick in a car.

Pump. The pump, as seen in fig. 2(e) is part of a few exertive tasks.

Wheel. There is a steering wheel, as seen in Fig. 2(f), mounted on the central stand. It can detect rotation, and allows the user to steer the submarine.

Valves. The two valves, shown in Fig. 2(g) are mounted on the steering wheel console and can detect whether they are in an open or closed state.

Pedal. The pedal (Fig. 2(h)) can detect pressure and is used in conjunction with other objects like gear box or the push button.

Pressure plates. The pressure plates, shown in Fig. 2(i) are used for balancing the submarine. The player steps on one of the plates, the balance of the submarine is then adjusted and fed back via the display of the underwater landscape.

2.3 Tasks and Complexities

The players' main task is to prevent the submarine from sinking. The players are given a certain time amount which elapses continuously, until the game is over. The user can increase the remaining time by solving tasks; not solving a task leads to time loss. The total number of concurrent tasks increases during the game so that the player will eventually not be able to complete tasks quickly enough to prevent the time from running out.

At the beginning of the game, the players are given a short series of tutorial tasks to familiarize them with the various smart objects in the room and their associated activities. These tutorial tasks are not intended to be stressful; the players are given as much time as they need to complete them. Once all tutorial tasks have been completed successfully, the actual game begins.

Even at the beginning of the actual game phase, the players are still learning how to play the game. While the objects in the room are now familiar to the players due to the tutorial, the actual tasks which they have to perform, are not all familiar to them (many tasks require interacting with multiple objects for instance). Early testing revealed that some players feel that a lengthy tutorial mode, which presents no challenge as such, is boring, so the remainder of the learning process was integrated into the main phase of the game itself.

Initially only a limited selection of relatively easy modules is present in the pool of modules which can be started. Whenever a task is started which has not been successfully completed by the players a single time yet, it is presented with an explanatory text in addition to the name and image associated with the task. When the task has been completed once, the lengthy explanatory text is left out, leaving the images, the name of the task, and in some cases a short description. The players are thus always presented with a selection of tasks, a few of which are new to them, but most of which are already known. This ensures that already learned tasks are not forgotten, that the players are not totally overwhelmed by a huge number of tasks which are unfamiliar to them, and it ensures continually increasing variety throughout the course of the game.

In general, tasks in the main phase of the game are designed to be very diverse to raise the game's overall complexity. Often more than one object is involved and the tasks' character reaches from being permanently in background ('avoid obstacles') over short-timed ones ('press the button 10 times') to tasks requiring object movement ('put the red box into the blue shelf'). Deliberately confusing tasks (*"do not press the button"*) are also started in order to increase cognitive load and further confuse the player. Due to the tasks complexity, sound is not used to announce them. Acoustic feedback however is given to indicate success or failure after the task ended. A tasks complexity is defined by various influences:

Task Announcement. Most of the tasks are presented as a box on wall with a short text and up to two pictures. These boxes "fall down" to indicate the task's deadline if any. The radio uses a beeper and its small display to announce radio-related tasks. Obstacles are announced by a calm but salient voice.

Cognitive Complexity. The amount of thinking required to understand and perform tasks varies from task to task ('enter a number sequence by gear' is much more difficult to accomplish than 'shake the multitool').

Spatial Extent. Some tasks involve no movement while some (e.g. 'shift gear then press button') require moving several meters or even make the player search for objects.

Duration. While some tasks can be solved in a sub-second interval some take a lot of time. Tasks with long duration can be further subdivided into long continuous actions (the pumping task) and tasks that do not necessarily require constant action but constant attention ('use pedal to keep speed in a specific zone', using the steering wheel to avoid obstacles).

Besides the tasks posed to the user, the system creates overload by several other means as well:

Locus. The locus of tasks announcements widens over time: Initially all tasks are presented in the center of the front wall; later on all of the wall space is used, as is a small display on the radio.

Parallelism. The number of parallel requests increases from one to about 8 during the course of a game.

Task Difficulty. The cognitive complexity of tasks increases: Initial tasks are very simple ("press the red button"), whereas later tasks require more processing and complex actions ("shift gear to 8, then press button", "do *not* press the red button", "hold the red box against all hull screws until the scanner shows red, then repair by clicking").

Time Pressure. Each task is given a deadline in which the user has to solve it. After deadline expiration the tasks is considered failed and the remaining time is decreased.

Immersion. Due to shape and size of the room and projective walls, the experience is designed to be immersive in the sense that the field of vision of the player is usually completely covered with tasks and game objects, while still ensuring an unthreatening experience by always allowing the player to naturally step backwards out of the room.

Audio. An increasing amount of alarm sounds, explosions, and countdowns further distract the player.

We will now briefly introduce our game tasks. Where not otherwisely stated, tasks will be announced via task announcement textboxes on the projection walls and have to be completed within a given timeout.

Switches. The radio will display a switch configuration which the player has to set up on the radio. This task is communicated only through the radio.

Pump. A bubble is displayed which gets bigger with pumping until it bursts.

Balance. The submarine is tilted to one side and the player has to step on the opposite pressure plate to rebalance it.

Shake*, *Shake2. The player has to shake one or two multitool boxes.

Charge Multitool. A multitool has to be hold against a power symbol on a specific position. The position is shown in the task announcement.

Listen. A specific multitool has to be hold to the players ear.

Steer Tasks*, *Shift and Steer. There will be different commands like 'steer to the left' or 'hold the steer' or 'engage gear X and use the steering wheel'.

Shelf. A specific multitool has to be put onto a specific place in the shelf, marked by different colors.

Authorization. Out of a set of authorization cards the correct one has to be chosen and placed on the radio. Additionally the switch configuration on the card has to be entered. This Task is solely communicated via the radio.

Radio Switches. The rotary knob on the radio has to be turned into a specific direction.

Radio Tune. The radio will display a relative signal quality. The player has to find the right rotary knob position to achieve a signal quality of 100%. This Task is solely communicated via the radio.

Gear. Different gears have to be shifted in the correct order.

Gear and Pedal. A gear has to be engaged while holding the foot pedal.

Button x10. The Push Button has to be hit 10 times.

Button Confusion. The player can do what he wants, but not press the Push Button. The resource manager will ensure that no tasks require a button press as long as this task is active.

Pedal. A gauge is displayed in the lower right corner of the front wall, the player has to control the pedal in order to keep the gauge in a defined range.

Power Failure. The walls will tint red and the user has to press a button to end the situation. Will not influence the time account.

3 Technical Background

In order to provide an immersive experience, Smart Objects in the panic room need to be able to robustly detect events without delay such that they be processed by the game logic and turned into visual feedback. Even slighly noticable delays or inconsistent behavior due to faults would immedietaly disrupt the game experince. In this section we present our Smart object hardware and the accompaning software stack that obeys these constraints.

3.1 Smart Objects

The interactive objects contain sensors and actuators, connected to an Arduino [Ban05] microcontroller platform. Each object contains a custom set of sensors such as accelerometers, buttons, light and pressure sensors and their respective electronic interfaces in addition to the battery. The location of movable objects can be determined using embedded RFID modules. Sensor data is preprocessed by the embedded device to ensure a reliable and responsive wireless connection to the server, using a fast, interaction-based protocol which is also used for hardware-control (e.g. writing data to a display or activating a beeper).

The low-level connection is established using wireless 2,4 GHz XBee modules using the 802.15.4 protocol [IEE11] in a point-to-point topology. Each object is powered by a lithium polymer battery, allowing runtimes of up to three hours.

3.2 Software Architecture

Communication. The micro controllers communicate with the server using custom middleware, which not only handles all of the communication, but also provides plug-and-play sensor management and failure handling. In order to provide quick reaction times and convey the feeling that player actions have an immediate effect, we developed a lightweight protocol, tailored for short response times. The protocol also provides auto-configuration of the smart objects and high reliability communications.

The protocol not only defines message types for sensor data, but also for maintenance purposes such as low power warnings or confirming that an object is still functioning. Each sensor type is managed by a driver which handles the sensor specific messages and provides sensor objects with a high level API to the game application. The server automatically uses the matching driver to create sensor objects, which can be used by the various game modules from then on.

When a game module requires sensor data, it requests a sensor object and registers a listener for the event it is interested in. The middleware transfers this listener to the micro controllers, which will respond as soon as the event occurs.

Error Handling. Since all of our smart objects are battery-powered, communicate wirelessly with the server and are exposed to the players, some failures and faults are inevitable. Player actions which remain unprocessed due to faults with the hardware would disrupt the immersive game experience and have to be detected and handled. If an Arduino fails to send three consecutive *Alive* messages, sent every 50ms, then the Arduino and all its sensors are considered temporally unavailable. Any running tasks which require the unavailable sensors are aborted, and modules which require those sensors can then no longer be started. As soon as the connection is re-established, the server notifies the Smart Object about the decisions it made due to the failure.

Resource Management and Scheduling. To avoid conflicting use of a single sensor device (e.g. by different tasks using the same object), tasks have to request the abstract sensor objects from the resource manager. Since tasks that only require a few sensors to run are more likely to get their resource demands fulfilled instantly, a scheduler takes special care of tasks with more dependencies, ensuring the desired balance of complex and simple tasks. If a task was not able to start for a certain amount of time because the needed resources were blocked by other game modules, the scheduler will correct the situation by not allowing other tasks to allocate the demanded resources. This way, the overdue task gets the chance to execute. The scheduler starts with a small pool of simple tasks, which is extended by new, more complex tasks one at a time, so the player does not have to learn multiple new tasks at a once in short succession.

Graphics. The graphical interface which is projected on the walls is implemented using the Java Processing framework [pro01]. Graphical objects are rendered in three different layers, dependant on their role in the game:

The topmost layer displays the task announcements. Each announcement consists of a box with text and images that explain to the player what needs to be done in order to solve the task. It appears at the top of the screen, falls down and has to be completed before it reaches the bottom.

The middle layer shows the interior of the submarine consisting of windows, control panels and other decorative elements. Its purpose is to simulate the submarine environment, but no direct user interaction with this layer is possible. A simple physics engine, based on verlet integration [Ver67], is used to gradually destroy the interior of the submarine. First the elements of the interior become partially detached from the 'walls' of the submarine, and start to swing with the movements of the submarine. Later they become completely free and move about within the submarine, according to its movement, to distract the players.

The background layer is rendered in 3D using OpenGL [SA94] and shows the underwater environment which can be seen through the windows of the submarine. The submarine is cruising through a landscape of hills that need to be evaded

by the player, to avoid time penalties and damage to the submarine. Besides the partially broken interior elements which are swinging around, the tilting horizon is the main visual clue of the rocking movements of the submarine. Tilting of the submarine is induced by interactions with the pressure plates, collisions with undersea mountains or special tasks that involve restoring the horizontal trim.

4 Observations

Data was collected by integrating a logger routine into the system for recording when the player succeeds or fails at a task. Additionally, players were asked to wear a pulse and blood oxygen analyzer in form of a fingerclip during gameplay which would allow us to get an indication of the physical effects of the game.

Players were also asked to fill in a questionnaire after having played the game. Capturing a video of each game (with allowance of the players) enabled us to analyze individual game situations and behaviors in detail.

4.1 Observations per Game

To get an idea of a typical run of a game we compiled the data from all the games into a single "average game" shown in Figure 3. The tutorial mode was cut out, an average number of active tasks was computed, and the pulse data was averaged into a single typical pulse curve. The increasing pulse shows, that using the Panic Room affects the players physical and/or emotional state to a degree that is measurable with a biomedical pulse sensor. Although the total number of tasks increases over the course of the game, the number of solved tasks stays constant, indicating the existence of an upper bound beyond which the players cannot perform better, despite increasing load.

Contrary to our initial assumptions, players did not primarily focus on the tasks appearing on the front wall. The screens haven been vertically divided into slots as seen in Figure 1. We recorded the failure rates of tasks depending on the slot they were announced in. The result is shown in Figure 4. The failure rate for slot 3 stands out notably low. This can be explained by the fact that the big red push-button, which is part of many tasks, is placed right in front of it and that most of the interactive objects in the room are within the range of a lunge, when standing right in front of slot 3. We thus suspect that players tend to focus on tasks announced close to their current position and will avoid moving.

4.2 Observations per Tasks

Figure 5 lists a ranking of the failure rates of the different tasks, depending on the phase of the game. We observe that the ranking of high-failing tasks changes drastically between the two game phases: For instance *Authorization*, one of the most complex and time-consuming tasks had a higher success rate in the later game. Steering tasks on the other hand seem to decrease in success rate. Since these two tasks are usually – due to the location of their respective

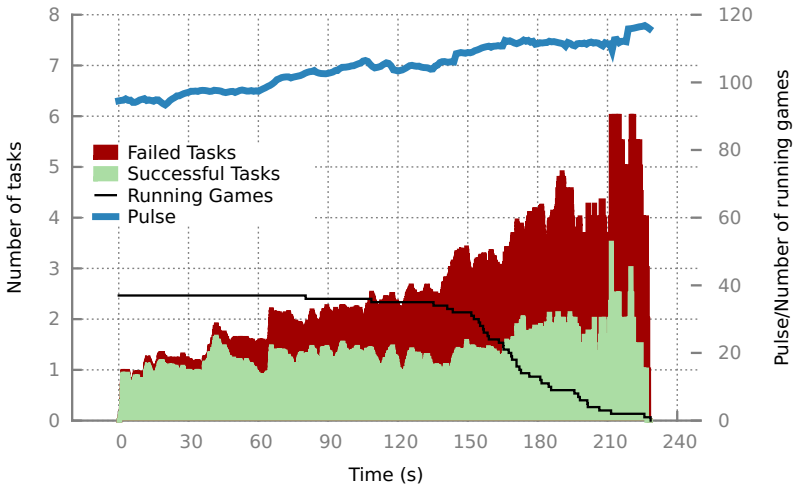

Fig. 3. An average game. The filled curves show the average number of visible tasks on the screen depending on the duration of the game. The portion of those tasks that were eventually solved is indicated in green. Not all games lasted equally long, thus data from later game phases averages over fewer samples.

objects – executed by the same person, this could indicate that the attention has shifted from observing the landscape and avoiding obstacles, to explicit task announcements. This impression is supported by the observation that the failure rates of the other tasks involving the radio have also become better or at the very least stayed similar, in spite of the higher overall task load.

The *Gear* and *Pedal* tasks increase in success rate, while the combined tasks using the gear has become worse. This could be explained by the fact that most of the combined tasks such as *Gear and Button* or *Shift and Steer* , where the objects are located remote, are done cooperatively, which becomes more difficult with increasing game speed.

The failure rate of *Button Confusion* has gone to zero, either meaning that it is ignored in a high-load situation or indicating a training effect on the players.

The Task *Listen* has become the worst of all although there are more complex tasks utilizing the multitools. This could indicate that the players perceive the task as more involving or demanding more commitment due to the incorporation of other body parts. It is also possible that the players are confused by the fact that although they are asked to listen, the multitool will not produce any sound.

We categorize tasks by “Task Announcement”, “Cognitive Complexity”, “Spatial Extent” and “Duration” as mentioned in Chapter 2.3. Figure 6 shows the resulting failure rates grouped by these categories. In the earlier phase of the game (before 2.5 minutes) when less tasks are active simultaneously a ranking of influences to the failure rate can be seen. It is most important that the task is communicated to the players in a clear manner. If they are not aware of the task, they are not able to perform it, and it is counted as “failed” in that case. The tasks’ content is less important: If they are easy to understand, do not last

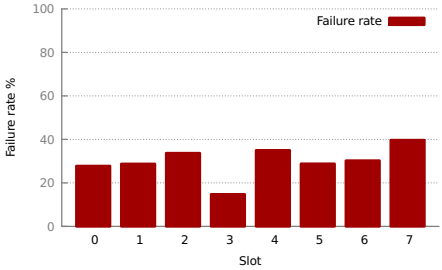

Fig. 4. Failure rates per slot during seconds 75 to 150. See Fig. 1 for slot placement.

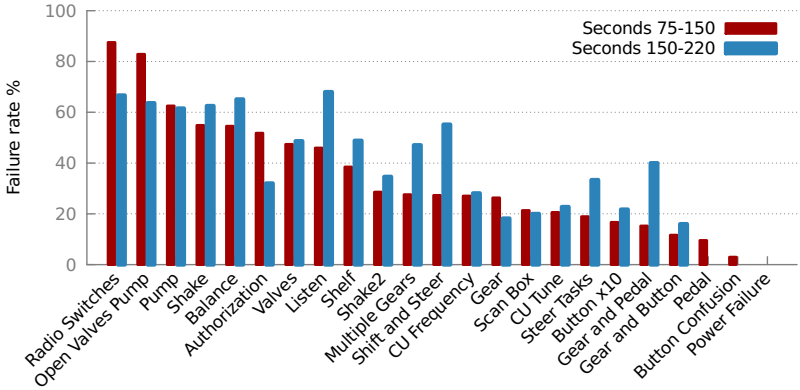

Fig. 5. Failure rate by task (second 75 - 150 and second 150 - 220)

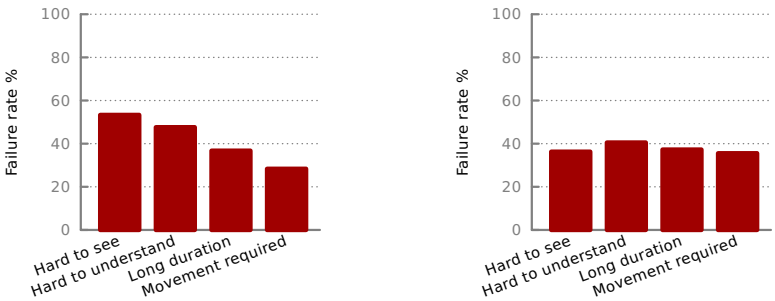

Fig. 6. Failure rates by category (second 75 - 150 and second 150 - 220)

too long, and do not require movement (in this order) their possibility of being accomplished increases. In the later game parts of the game (after 2.5 minutes) the influence of these factors on the failure rate seems to be almost equal. This is caused by the fact that players get to know the tasks and learn how to solve them efficiently.

5 Conclusion

In this work we presented the "Panic Room", a versatile platform that allows to observe the reaction of users when overloaded with interacting with ubiquitous systems in a playful setting. In our first set of experiments we could already draw several conclusions about a typical player behavior such as the average player being able to solve tasks at a constant rate independently from the load by ignoring spatially remote tasks. Furthermore we could show the importance of task descriptions right in front of the users and how more complex tasks can be learned in the course of a game. In further studies the role of acoustic feedback for the user experience in overload situtaions could be evaluated in more detail. Due to its extensibility, we believe the Panic Room can be a valuable instrument for future AmI studies.

Acknowledgments. This work was partially supported by the European Union under contract number ICT-2009-258885 (SPITFIRE). The Panic Room received partial funding from student tuitions.

References

Ban05. Banzi, M.: The arduino microcontroller platform (2005), `http://arduino.cc/`

Dem08. Demiris, G.: Smart homes and ambient assisted living in an aging society. New opportunities and challenges for biomedical informatics. Methods Inf. Med. 47(1), 56–57 (2008)

Fog02. Fogg, B.J.: Persuasive Technology: Using Computers to Change What We Think and Do (Interactive Technologies), 1st edn. Morgan Kaufmann (December 2002)

GAL. Lower Saxony Research Network Design of Environments for Ageing, `http://www.altersgerechte-lebenswelten.de`

IEE11. IEEE. IEEE Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Std 802.15.4-2011 (2011)

KMRA09. Kaptein, M., Markopoulos, P., Ruyter, B., Aarts, E.: Persuasion in ambient intelligence. Journal of Ambient Intelligence and Humanized Computing (1), 43–56 (2009)

pro01. The Java Processing Framework (2001), `http://processing.org/`

SA94. Segal, M., Akeley, K.: The opengl graphics interface. Technical report, SILICON GRAPHICS COMPUTER SYSTEMS (1994)

Ver67. Verlet, L.: Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. Phys. Rev. 159, 98–103 (1967)