NO ONE LEFT BEHIND

D.4.2 - Pocket Code functional specification framework and integration of transferred technologies (Final Version)

Grant Agreement number: 645215
Project title: No One Left Behind
Funding Scheme: Innovation Action

Due date: 30/06/2017
Actual date: 24/07/2017
Document Author/s: Bernadette Spieler(TU Graz), Anja Petri(TU Graz), Maria Eugenia Beltran (INMARK), Patrick Münster (HdM), Eugenio Gaeta (UPM), Teofilo Redondo (ZED)
Version: 1.1
Dissemination level: PU
Status: Final Version

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645215

Document History			
Version	Date	Comments	Author
0.1	22/11/2015	Creation of the document and first draft	Bernadette Spieler(TU Graz), Anja Petri(TU Graz)
0.2	30/11/2016	Inputs to first draft	Maria Eugenia Beltran (INMARK)
0.3	05/12/2016	PMD additions	Eugenio Gaeta (UPM)
0.4	09/12/2016	GPII Integration	Patrick Münster (HdM)
0.5	20/12/2016	Analytics and Dashboards information	Teofilo Redondo (ZED)
0.6	21/12/2016	Inputs to 5th draft, submission D4.1	Bernadette Spieler(TU Graz), Anja Petri(TU Graz)
0.7	21/12/2016-15/01/2017	Inputs, restructuring and integration of new content (D4.2 final version)	Maria Eugenia Beltran (INMARK), Bernadette Spieler(TU Graz), Anja Petri(TU Graz)
0.8	19/01/2017	Review changes	Bernadette Spieler(TU Graz)
1.0	26/06/2017	Inputs, restructuring and integration of new content (featurese, templates etc.)	Bernadette Spieler(TU Graz)
1.0	24/07/2017	Final edits	Bernadette Spieler(TU Graz)

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GPII	Global Public Inclusive Infrastructure
MDA	Mechanics, dynamics and aestetics
PMD	Project Management Dashboard
UI	User Interface

EXECUTIVE SUMMARY

This deliverable presents the Create@School app, which is the inclusive and ludic enhanced New Generation of Pocket Code. In combination with the Project Management Dashboard (PMD) an environment for teachers, it provides value and empowerment for education. This document considers the functional conceptual framework as defined in Delivery 1.2 and the Create@School training guide tools developed in Delivery 2.4. The different parts of the Create@School app are descripted seperatly and listed shortly below:

Game oriented modules (templates): A key element was to integrate a game-making teaching framework (GMTF) into Pocket Code. The concept of the NOLB GMTF is based on principles of the Universal Design for Learning (UDL) model, see Delivery 2.4. Its focus lies on three pillars of learning: the *what*, *how*, and *why*. Thereby, the NOLB GMTF is a common set of concepts, practices, pedagogy, and methods. This framework provides a coherent approach to learning and teaching by integrating leisure oriented gaming methods into multi-discipline curricula. One output of this framework was the integration of game-based methods via game templates that refer to didactical scenarios that include a refined set of genres, assets, rules, challenges, and strategies. The pre-programmed, re-usable templates were adapted to learning scenarios for the NOLB experimental pilots. They allowed: 1) teachers to start with a well-structured program, and 2) pupils to add content, adjust the code to integrate their own ideas and it helped them to follow important design guidlines or to stick to coding principals. During the project game genres such as adventure, action, and quiz, as well as rewards or victory point mechanisms, have been embedded into different subjects, e.g., science, mathematics, and arts. The insights gained during the class hours were used to generate 13 game templates, which were integrated in Create@School.

Project Management Dashboard for teachers: The developed PMD helped teachers to manage their classes, students and projects. In addition, it provides opportunities to evaluate students programs with the use of different parameters. For this, cognitive and behavioural measurements in No One Left Behind has been handled through two data collection processes:

- teachers observation and assessment through the PMD, e.g., academic readiness and achievement of learning goals (learning achievement)
- automatic collection of data from the use of the Create@School app by students (e.g., program design, usage of different coding concepts) gathered by an SDK within the source code of the app

Therefore thee methodological framework is part of Delivery 4.3.

Analytic tool: Create@School tracks different user input / events while students are programming, e.g., when they creating or deleting objects, time spending with playing or drawing in Pocket Paint etc. In this way teachers can follow the process of thinking and collaboration and discover key learning experiences and best practices. This helps to control learning activities in game-making, to see gaming insights and adapt lessons plans, to tap into intrinsic motivation and increase engagement and pupil retention. Visualistaion dashboards have been provided online (big data software Tableau) and offline (excel sheets).

GPII inclusive technology: In Create@School students or teachers can choose a preferred profile to support students with special needs such as dyslexia, cognitive or visual impairments.

Create@School has been tested and evaluated in the three pilot countries (UK, Spain, and Austria) during the second cycle of the project (see Delivery 5.3, 5.4 and 5.5).

1 INTRODUCTION

WP4 aims to bring the new app Create@School towards a near market ready version. Create@School is the New Generation of Pocket Code. Therefore, it is Pocket Code based enhanced for the educational environments and embeds the concepts of game mechanics and dynamics from leisure oriented video-game companies through ready-to-use (pre-coded) game templates, as well as the GPII inclusive framework.
With Create@School students can program games and create animation their way. Both, students and teachers can easily create their own interactive and dynamic games, animations or digital stories using daily life and academic content. Create@School provides a media-rich programing environment, with ludic enhanced coding possibilities, that responds formal academic primary and secondary contexts. With the activities of the WP we connected Pocket Code with the same technological base and services as the ones used in "traditional gaming market", GPII inclusive technology and linked it with the formal academic and pedagogic CVs, as well as teachers 'practices, gathered in WP1 and WP2.

The game templates comprise different types of games coming from the leisure gaming environment; including adventure, action or puzzle games as well as rewards or victory point's approaches. These templates are prepared (coded) in Pocket Code; which are a catalyst for games that can be easily linked to the different academic competences for different subjects and classroom ability levels.

Our aim was to enhance students' abilities across all academic subjects, as well as their computational proficiency, creativity and their social skills. Through the first periods of the project, Pocket Code has been validated as an effective earning and teaching tool. On the one hand, Pocket Code already helped its users to learn basic programming steps but on the other hand, the app was difficult to transfer to other subjects besides computer science. The app needed improvements in terms of usability by decreasing both complexity and the time needed to develop games. Therefore, a more powerful and usable interface for the management of objects and large programs has been integrated into Create@School, as well as further features and frameworks (GMTF).

This enhanced version, with the name Create@School has been developed during the first cycle and uitilzed as an open beta version to our pilot schools during the second cycle of the project which started in October 2016.
In this regard, the new functional framework that holds Create@School is illustrated in Figure 1, and explained in the following chapters.

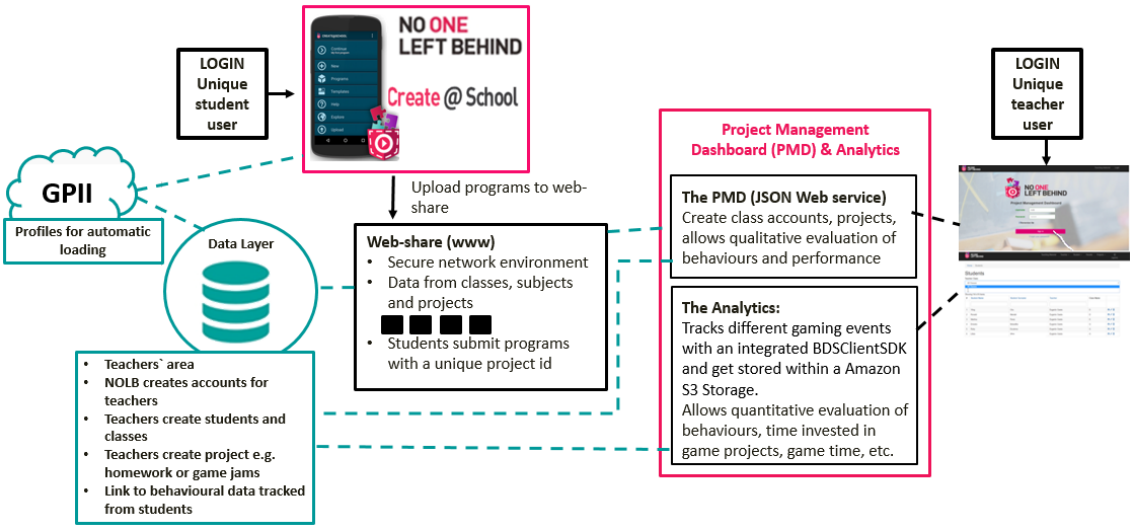


Figure 1: The overall framework of Create@School

Moreover, we developed new practices to improve the setup of the whole courses and new way how to integrate Create@School in the classes. The results wre embedded in the Teacher Framework as well as in the Project Management Dashboard (PMD). The PMD allows to orchestrate the learning environment where the teachers create "projects" (homework or activities) to support teaching and strategies for creating class material and learn through games. Linked to the PMD the analytic tool allows to follow the process of thinking, collaboration, the qualitative completion of the goal pursued and the students' behaviours (times spent creating and playing the games). The tool allows teachers to integrate a new pedagogy in a scaffolded manner and tracks behaviours to engage and improve innovative teaching-learning approaches.

Thus, the functional specification framework for NOLB comprises two components, which are represented in Figure 1:

- The Create@School App
- The Project Management Dashboard and the Analytic Tool

2 CREATE@SCHOOL APP: THE NEW GENERATION OF POCKET CODE

Create@School is an enhanced version of Pocket Code. This version integrates the results of our observations during the pilot studies and considered feedback from teachers and students. Furthermore, new infrastructural services were developed, e.g., data tracking of certain events or certain times during game creation. This data has being analysed to create statistic information and visuals of students´ behavioural parameters. We also enhanced our e-learning (edu-) platform for teachers to cooperate in the creation of lecture specific material. Another important feature was the integration of different pre-coded templates and also nesting objects as object collections (grouping several objects) or creation of levels through scenes. Finally, the app also supports a set of accessibility features, which optimise the app also for students with special needs.

To summarize the New Generation of Pocket Code consists of different parts:

- the new Create@School app, which is the school oriented version of Pocket Code created under the NOLB project:
  - it is an extension of the Pocket Code application for teaching other subjects than only object oriented programming
  - facilitation of Pocket Code programming and reducing the time needed in order to develop games and applications
  - development of 48 new features and improvements (see Annex I to see the list of these features and improvements) performed during the project and based on the experience and insights gathered from the NOLB pilots, which have used Pocket Code during the 2015-2016 scholar year. In May 2015, a list of 67 planned and forthcoming Pocket Code improvements were suggested by the NOLB consortium. During the project, we were able to release 76% of these features and integrate them into the Create@School app. As a result, the app is now almost Scratch complete. Until the end of the project only less features will be integrated because the team mainly focuses on code refactoring, clean code and test refactoring.
- the Project Management Dashboard (PMD) for teachers
  - account management: managing the submission process of Create@School programs.
  - programs are submitted through the web-share with a unique project id. Therefore a button has been integrated within the program overview of the students' accounts.
- the collection of analytic data
  - monitoring of certain activities of Pocket Code (Pocket Code/Analytics)
  - monitoring time while playing a Pocket Code game (Games/Analytics)
  - monitoring activities during the game making process on specific subject in school (Game making/Analytics)
- the predefined templates
  - to optimise design choices, particularly in the area of game mechanics and player feedback systems
  - integration of ludic oriented game mechanics has been embedded through game templates (i.e. templates for adventure, action or puzzle games). The templates and the Create@School app also enable game dynamics by editing an existing game design but allowing personalisation of backgrounds, landscapes, characters, the creation of new challenging levels, as well as changing the difficulty of a game.
- set of accessibility preferences
  - making the Pocket Code "IDE" app more accessible

- integration of GPII framework that allow automatic and individual personalization.

These parts are summarized in Figure 2.

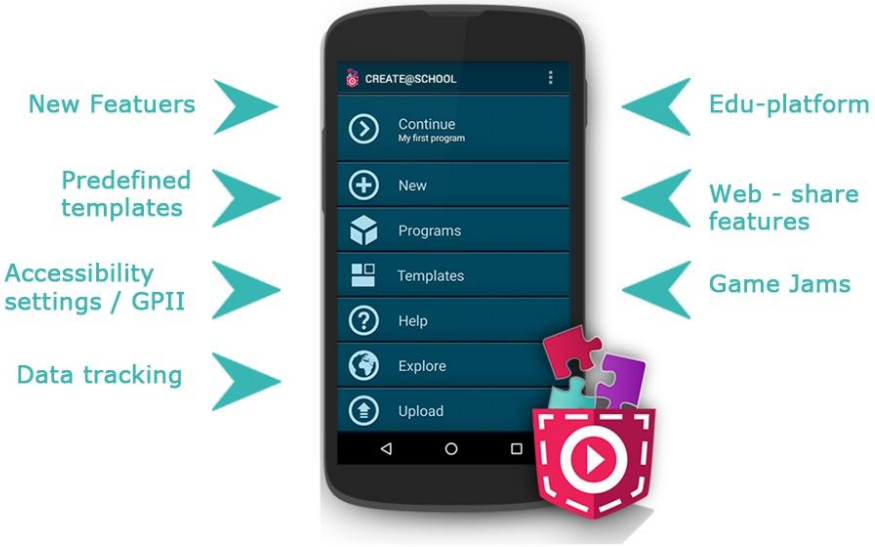


Figure 2: Create@School functionality framework

2.1 The Create@School App

For the second cycle of the project (October 2016) the first prototype of the Create@School app has been introduced in schools of our piloting countries (UK, Spain, and Austria). The app was improved and released seven times during the second cycle. With every iteration Create@School got closer to a maket ready application. Table 1 holds a description of every version.

Version number	Release date	Tasks
2	30.10.2016	Spanish translations, issues with clone object, copy scenes, look references, special characters, ...
3	01.12.2016	Size of template icon, issues with: Spanish strings, physical object collision, backpack, scene (local variables), login dialog
4	15.12.2016	Wrong escaping of scenes' names, bug in quiz/puzzle template
5	26.02.2017	New features: - Ask brick with spoken answer - 5 new templates: o Action platform, Action shooter, Interactive Book, Physical simulation, EV3/NXT simulation
6	04.03.2017	GPII improvements - Apply fontsize in formula editor with respect to device - Resize compute dialog - Apply the font face for hints - Add bricks to "Simple bricks" - Uniform icon size across devices - Apply profile always in main menu after restart - Change color scheme for hints New events to track: - programID - hints enabled, broadcast messages
7	30.06.2017	Last 4 templates: - Adventure RPG - Race simulation - Life simulation - Strategy
8	August 2017	Two GPII improvements: - simple bricks

Version number	Release date	Tasks
		- FE Intro Chromecast feature Bug fixes (clone brick, stop all scripts, bubble bricks)

Table 1: Create@School releases

Create@School was released as an open beta test apk. Open beta testing allows to run a test with a large group and surface the app's beta version on the Play Store. When running an open beta test, anyone can join the beta program with a URL link and submit private feedback. The URL link was shared with our teachers via email. Each tester needs to opt-in using the link to be part of the beta test, see Figure 3. The open beta version of the app has no public ratings and you need a nolb account in order to log in.

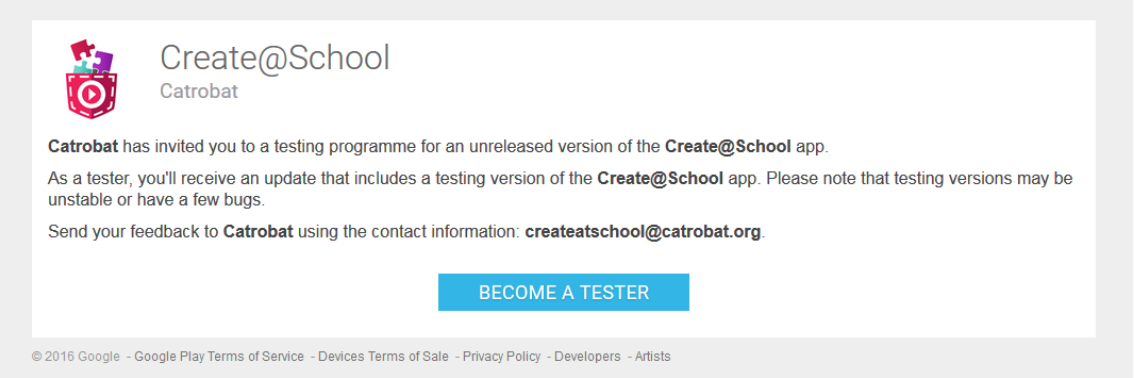


Figure 3: Notification to become a tester

The login for the app is required to bind the tracked data to one username. The NOLB team created 600 accounts per school. The username has the following sequence:

- nagif0001
  - n = NOLB, a = Austria, gi = school code, f/m for female and male and a sequential number

In addition three teacher accounts per school have been created.
This sequence ensures that the data of the students is anonymised. To also identify the students' grade every class starts with a certain number e.g., 5th grade 0001 – 0049, 6th grade 0050 – 0099 etc. Figure 4 shows the beta app in the Google Playstore and Figure 5 pictures the new launcher icon.

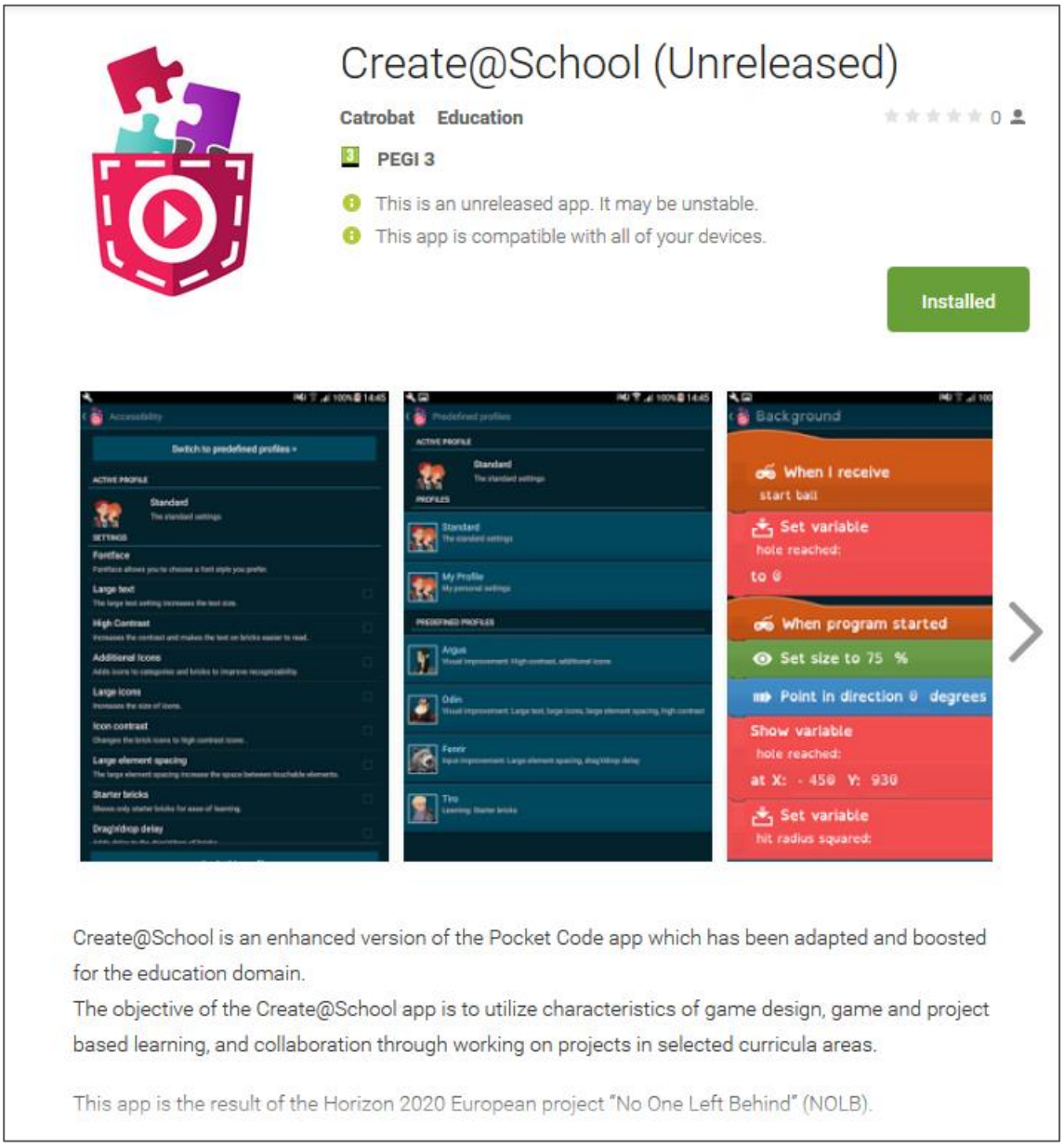


Figure 4: Create@School in Google Play

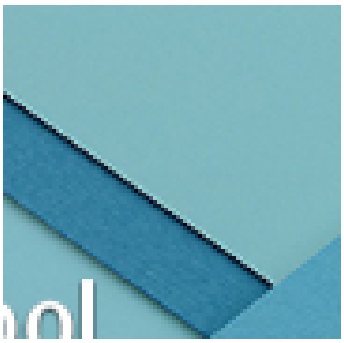


Figure 5: New launcher Icon of Create@School

2.2 New and incremental developments that shaped Create@School

In T1.6 we gathered functional specifications and components to conceptualize the New Generation of Pocket Code, considering the inclusive features to make Create@School a ludic and inclusive oriented app that enables game based teaching-learning approaches in classes.

New and incremental developments can be divided into the following main areas:

1. Integration of transferred technologies: Generation of templates. Adding and development of features and functionalities
2. Validation of templates
3. GPII Integration
4. Enhancement of users 'experience and usability
5. New functionalities to link with Scratch
6. New hardware extensions
7. Support of new operative systems/environments

2.2.1 Integration of transferred technologies: Generation of templates

Under this approach, we have integrated ludic oriented game mechanics which has been embedded through game templates (i.e. templates for adventure, action or puzzle games), as well as rewards or victory points approaches. The templates and the Create@School app also enable game dynamics by editing an existing game design but allowing personalisation of backgrounds, landscapes, characters, the creation of new challenging levels, as well as changing the difficulty of a game.

Create@School has now re-usable templates/modules for projects e.g., in history, sciences, physical education and physics. The NOLB templates have been created by adapting game-modules, which have been described in D 2.1, by means of the customization of the Pocket Code Gaming Components depending on game genre, see Figure 6.

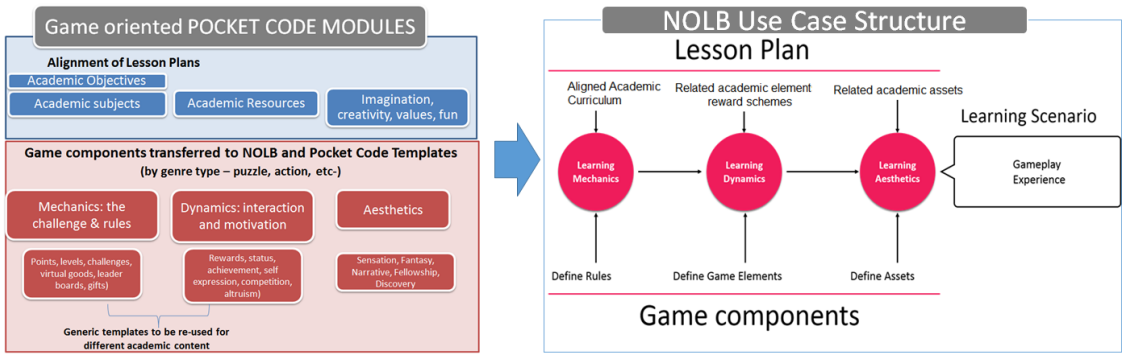


Figure 6: Game oriented modules

The game-modules were developed based on different types of games (game genres) including Quiz, Adventure, Puzzle and Action games. In the 1st cycle, as planned, four game-modules has been developed. For the 2nd cycle (during the second year of the project) nine more game-modules have been developed (five until February 2017 and the last four until June 2017), based on sub-genres of action, adventure, puzzle and quiz.

Through these templates, Pocket Code easily supports the creation of games that replicate the mechanics of the ones used in leisure environments, with the creation of a framework structure (template) that guides the development of the game.These templates 'configuration are described in D.2.4.

A new option "Templates" has been added to the Create@School's main menu to access the templates list, see Figure 75.

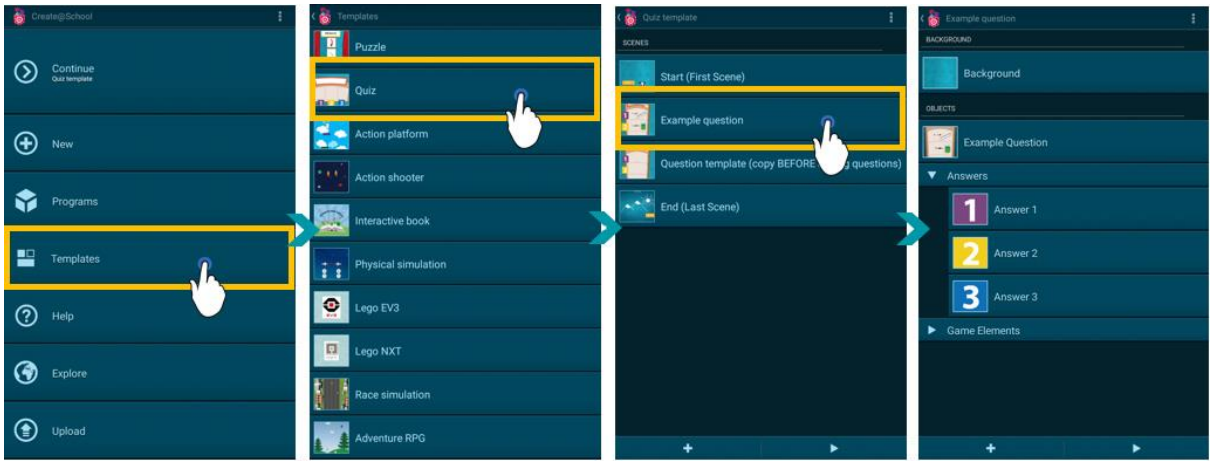


Figure 7: Game templates in Create@School that supports scenes and grouping of objects

By changing or adding different contexts, game assets, or game mechanics the pupils adapt, customise, and create diversity with the dynamics and aesthetics of the games. By using the game design elements they can build new games and remix existing ones. The templates allow to edit the existing design, giving the pupils freedom in the personalisation of backgrounds and characters. Fun, engaging experiences are generated through the creation of new, challenging levels or changing the difficulty of a game. The table below shows all developed templated, their underlying subject, their theme and learning goal. In addition, they are linked to the MDA framework.

Game mechanics, dynamics, and aesthetics (MDA) provide a consistent structure to define game elements, goals and rules and thereby deliver a common framework and vocabulary for games [10]. The MDA, explained in Delivery 3.2 is a formal approach to understand games and their elements in order to support the process of designing and developing a game [9]. *Mechanics* are a synonym for the "rules" of the game and defines the objects, elements, and their relationships. They comprise points, levels, challenges or virtual goods. *Dynamics* describe the play of the game when the rules are set in motion. They comprise rewards, status, achievement, self-expression, or competition. *Aesthetics* refer the player's experience with the game. They are the reason for playing games and comprise, for example, fantasy, narrative, fellowship, and discovery.

Genre	**Subject**	**Theme**	**Learning goal**	**MDA**
Quiz	Physics	Properties of physical objects	Learn about physical objects and their properties (e.g., inertia) through questions and answers.	points, knowledge, narrative
Adventure	Science	Space	Listen to a space scenario and decide "Yes" or "No".	levels, discovery, narrative
Puzzle	Music	Instrument groups	Tap on the musical instrument, which does not fit to the group (odd one out).	points, timer, logic rules, narrative

Action	Science	Respiration	Learn about de-/oxygenated blood cells by tapping on the objects. Avoid the virus cells.	points, timer, high-score, fellowship
Action Platform	Physics	Periodic system	Move the character from surface to surface, to catch halogens.	Points, timer, narrative
Physical Simulation	Physics	Newton's laws of motion	Perform physical experiments with Newton's 2nd law of motion and experiments with the formula.	Submission, expression, discovery
Action Shooter	Maths	Division rules	Shoot asteroids which can be divided by 4, 3 and 11.	levels, points, timer, narrative
LEGO Simulation	Computer Science	Robotics	Use the LEGO NXT / EV3 extension and solve tasks (e.g. creating a maze). Learn about sensor values, or coordinates.	levels, submission, challenge
Interactive book	Science	Water cycle	An easy version of an interactive book with scenes that explains the water cycle.	levels, submission, expression
Adventure RPG	Fine Arts	Colour circle	The user has to collect colours (inventory) to draw a picture. Additional: Create your own character.	levels, virtual good, achievement, challenge
Racing simulation	Science	Pollution	Collect trash to get points and level up. Avoid the other cars to keep playing.	levels, points, challenge, achievement
Life simulation	Life after school	Work skills	Help the character serve the right order and keep the customers happy.	levels, points, challenge, achievement
Strategy	History	eras (greek, egypt, romans, kingdoms)	Multiplayer (on one tablet). Strategy decisions via a simple connect 4 template.	Points, leadership, narrative

Table 2: Overview of the developed game templates

Almost all templates are available in portrait and landscape mode and have be translated via crowdin[1] in the languages of the pilot countries. Instructions to the templates are found on our NOLB edu-platform, see https://edu.catrob.at/no1leftbehind-for-teachers. For a better understanding, the app provides one example (subject) level of each game template in order to allow also sense of progress and challenging creative options.

The first four templates were directly stored in the app. To ensure a continuous extension, exchange and improvement of the templates and to avoid that the app gets to big we outsourced them in Feburary 2017 and they are now available for download directly in the app.

As part of this task, also specifications to determine "validity" of scenarios and metrics were approached (relation and need of realistic graphics, required fidelity and goals needed to be achieved) through the school year 2016-2017. In this regard the game templates include adventure, action or puzzle games as well as rewards or victory point's approaches, see Table 2. These templates are prepared (coded) in Pocket Code; which are a catalyst for games have been selected by pilot`s users as easy to be linked to the different academic competences for different subjects and classroom ability levels.

Deliverable 1.2 defined the parts of the "Shape of a Game" ceremony and important characteristics about each genre which which have been explained in more detail in Deliverable 2.2. The templates refer to the MDA framework to transfer the mechanics, dynamics and aesthetics of games. These rules for a game helped us to shape the templates for the Create@School app. Important game mechanics (see D2.2) have

[1] Localization managment paltform:https://crowdin.com/project/catrobat

been considered like the possibility of earning points or adding different levels as well as game dynamics (e.g., achievements, self-expression, competition).

To use the same shape[2] for all game templates, the following structure has been developed:

- Title screen
- Instructions screen: Conveys "goal" and "rules".
- One or more levels: Use of the word "level" creates a connection to commercial games.
- An end screen: This is linked to the end of a story, the achievement of a target, etc.

The genres in Figure 8 have been identified as offering useful models for gameplay [11].

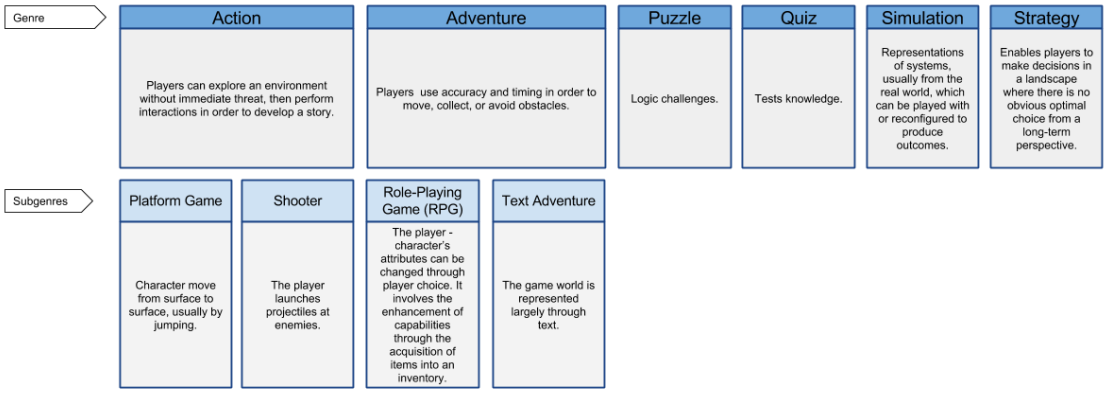


Figure 8: Clustering of game genres in NOLB

The game genres helped to define what game design elements are necessary to effectively create the chosen genre and that the theme 'fit' into the genre classification. By adopting learning content into something that appears to be a game, a new experience for pupils is created.

All templates have important elements in common:
All templates

- start with a game screen (in different colors), followed by an instruction screen and end with end screen, see Figure 9.

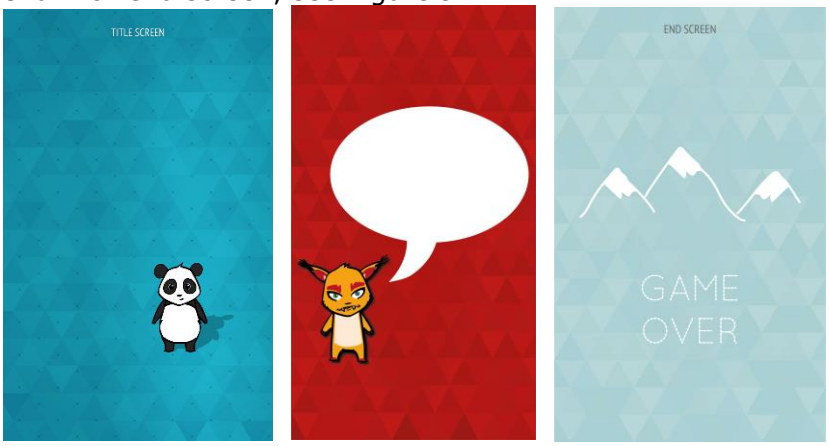


Figure 9: The "Shape of a Game" ceremony

[2] The project drew upon the work of partners' visiting speaker Gary Penn, formerly Creative Director at DMA Design (maker of the original Grand Theft Auto game).

- contain an example level that fits to one subject to show the user the game play, e.g. biology, music or physics
- have a template level integrated to be modified by the user
- are easy to expand and provide the opportunities for self-expression e.g., users could choose their own graphics, modify the "Shape of a game" – screens and play around with the different game elements
- use different HUD elements like points or a timer to reward the users actions
- have at least two levels integrated which displays different milestones that a player must achieve and provide hence a new challenge for them
- have a logical structure
- are devided into different scenes and object groups.

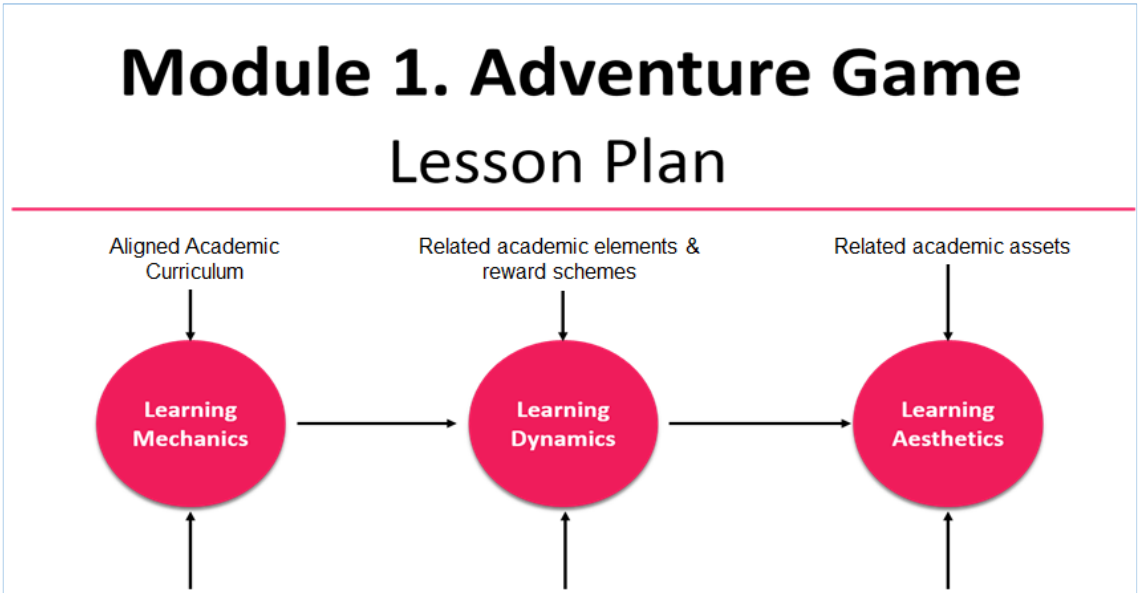
In addition, we provide a game template guides for the teachers3. These guides represent a step-to-step tutorial and show how to use each template. With the help of these guides the teacher then can decide which of the templates fits best to his or her subject and the learning goal within the curriculum.

The challenge was on the one hand, to pre-program the templates in an efficient way which provides students enough functionalities so that it is easy for them to start with a new game and also explains them important dynamics, mechanics and aesthetics of a game (like coding how to reward the achievements or allowing to collect points). On the other hand we want them to have the freedom to express themselves in a creatively, fun and fascinating way (e.g. allow them to change images, sounds) and to show them different ways to archive the goal which all supports their logical thinking processes.

In the following all templates are descripted again in more detail.

2.2.1.1 First cycle of templates: October 2016

Module 1: Adventure Template
This module focuses on exploration. Our template involve a simple puzzle solving. The user's progress depends on understanding the dialogues and actions of the game. Figure 9 shows which mechanics, dynamics and aesthetics were actually transformed to the action game.



[3] https://edu.catrob.at/no1leftbehind-for-teachers

Requirement components: none **Optional components**: levels (3 example levels), challenge during questions **Rules**: Explore the different levels	Required components: none **Optional components**: achievements: the next level is unlocked when complete the first one, self-expression: use own graphics, create own story	**Required components**: narrative – try to reach the end of the game **Optional components**: sensation, fantasy, challenge, fellowship, discovery, expression, submission

Figure 10: Use Case structure of implemented adventure game

Adventure Guide: Game Play: The adventure template contains five scenes:

- Start (First Scene)
- Example Level 1
- Example Level 2
- Template Level
- End (Last Scene)

Figure 11: Scenes for the adventure template

This game is a linear adventure game. This means there is only one way to get to the goal. You play by answering different questions and have to give the right answer to get one level further to the goal. The game starts with the first level (image 1). By answering with NO the game will be over for you (image 2). For the second level you have to answer another YES/NO question (image 3). When answering with YES the game will be over for you, if you answer with NO you get to the third level, the user will get a short explanation why his answer was correct (image 4). By answering with NO you get to the third level (template level) (image 5).

Image 1 Image 2 Image 3 Image 4 Image 5

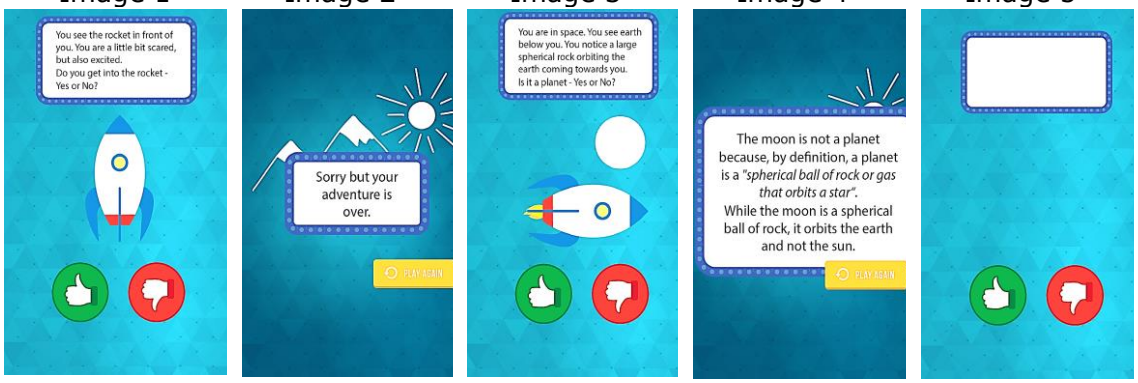


Figure 12: Sequence of the Adventure game

Module 2: Action Template

Action games centre on the player and emphasize challenges that require eye-hand coordination. Figure 13 shows which mechanics, dynamics and aesthetics were actually transformed to the action game.

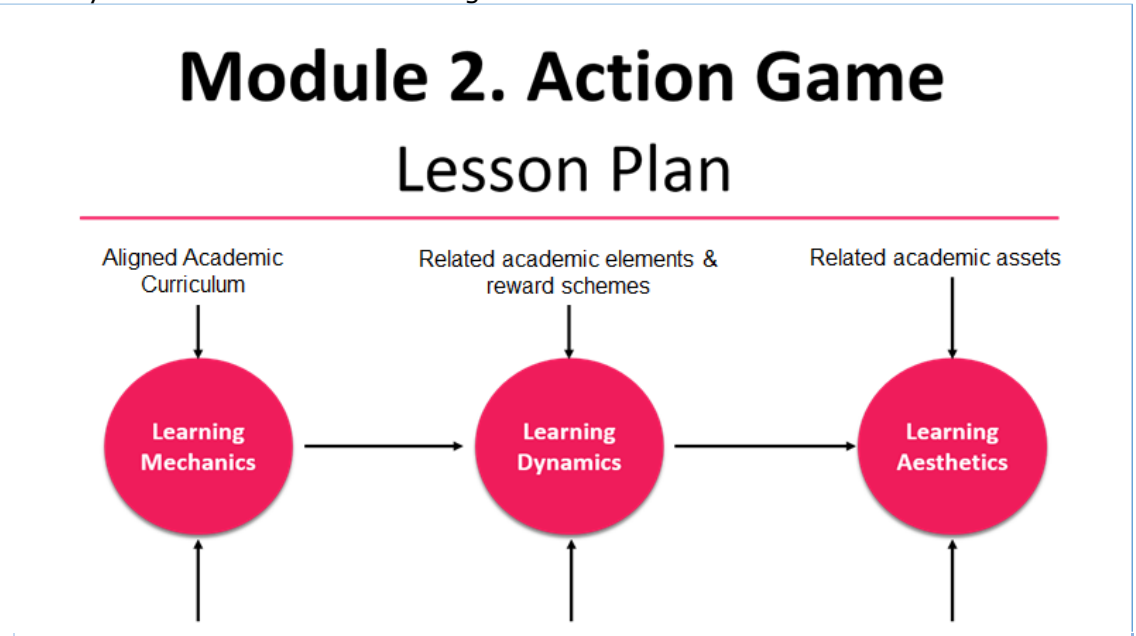


Figure 13: Use Case structure of implemented action template

Game Play: The Action template contains four scenes:

- Start (First Scene)
- Question (Add a game question)
- Action (add looks to tap on)
- End (Last Scene)

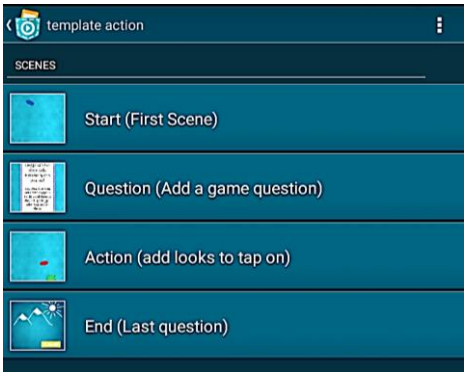


Figure 14: Scenes of the Action template

At the beginning of the game the user will get an info screen that tells him what he or she is allowed to tap on. During the game the different HUD-elements are shown – a timer, high score and the actual score display. A HUD is short for **H**eads **U**p **D**isplay and is a display area where gamers can view their characters vital statistics (such as health), and can indicate game progression (such as score). What attributes are shown in the HUD depends on the game. While the information that is displayed on the HUD depends greatly on the game, there are many features that players recognize across many games. Most of them are static onscreen so that they stay visible during gameplay. The user has 30 seconds (timer) to catch as many blood cells as you can. With each one you catch, the score increases by 1. The high score is saved with each restart (restart button at the end of the game) and you can try to improve on this each time. Avoid the virus cells – if you catch one of these the virus cell increases in size and your score decreases by 1. The sequence of the game is shown in Figure 15.

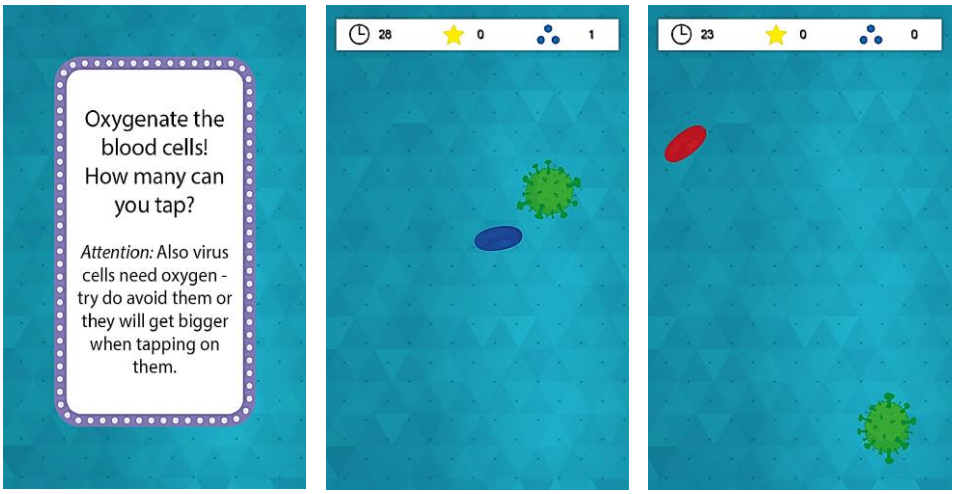
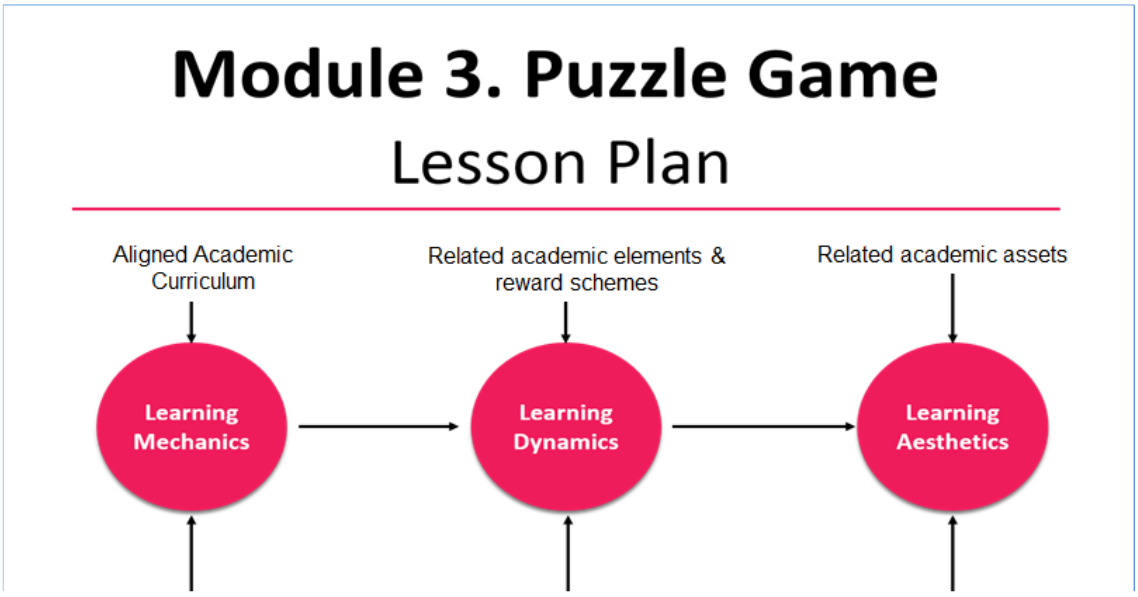


Figure 15: Sequence of the action template

Modul 3: Puzzle Template

Puzzle games are logic games where the player solve puzzles. Our template run against time and follows the principle of "Odd one out". Figure 16 shows which mechanics, dynamics and aesthetics were actually transformed to the puzzle game.



Requirement components: none **Optional components**: points when the right item is chosen and a timer (3 seconds) **Rules**: logic challenge rules	Required components: none **Optional components**: rewards when choosing the right item to odd out, self-expressing when choosing own graphics	**Required components**: narrative – try to answer all challenges correctly **Optional components**: sensation, fantasy, challenge, fellowship, discovery, expression, submission

Figure 16: Use Case structure of implemented puzzle template

Game Play: The puzzle template contains three scenes:

- Start (First scene)
- Puzzle (add your looks)
- End (Last scene)

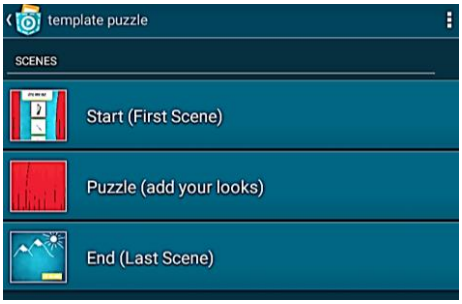


Figure 17: Scenes of the Puzzle Template

At the start of the game you will see a red curtain (Image 1). By clicking on it, it opens and the game starts. This example shows you three musical instruments. One of these instruments does not belong to this group (Image 2). In this example the two correct instruments have a green border. The wrong one is highlighted by a red border. The timer at the top shows you the time you have left choose. If you need longer than three seconds, or if you choose the wrong image you lose a point (score minus 1). If you choose the correct image, your score increases by one (Image 3). The order of the images changes with every game start.

Image 1 Image 2 Image 3

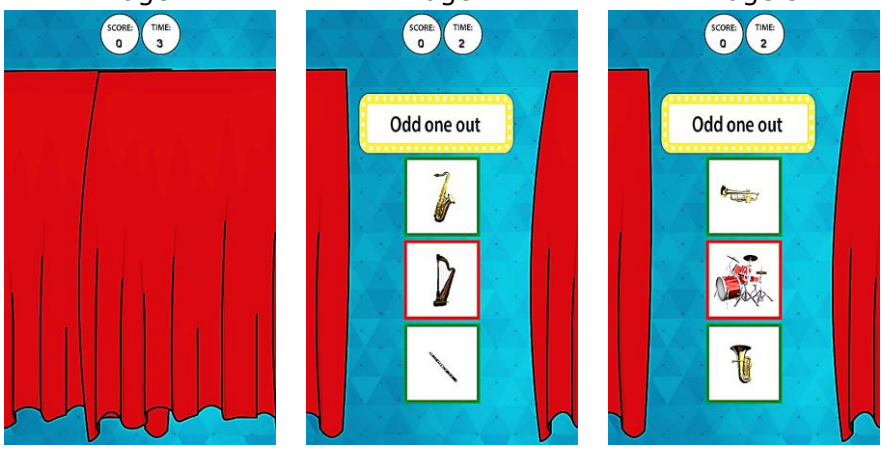


Figure 18: Sequence of the Puzzle Game

Modul 3: Quiz Template

In quiz games the objective is to answer questions with the goal of obtaining some rewards (points). Figure 19 shows which mechanics, dynamics and aesthetics were actually transformed to the puzzle game.

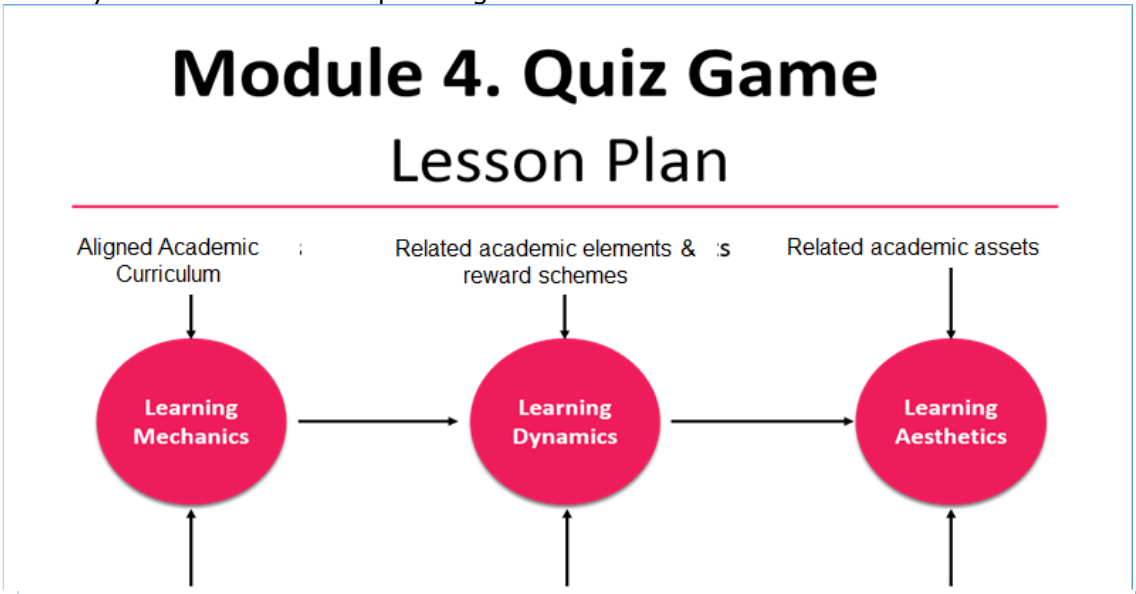


Figure 19: Use Case structure of implemented quiz template

Game Play: The quiz template contains four scenes:

- Start (first scene)
- Example question
- Question template (copy this question before creating your own)
- End (last scene)

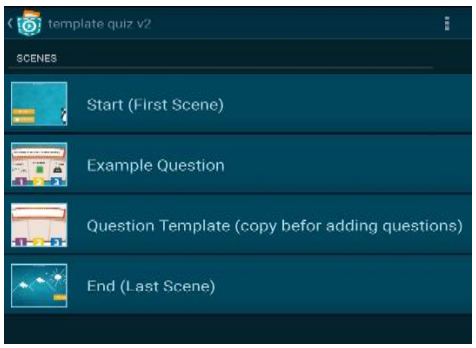


Figure 20: Scenes for the quiz template

Each question scene ask one question and provide three answers. In addition the player could listen to the answer by clicking on the recorder icon. The player choose

the answer by clicking on one, two or three. A visual feedback either a check mark or and red X shows if the question has been answered correctly. In a second step the player gets additional information to the question. With the button "Next question" the next question appear. With every correct answer the points change by 1. The sequence of the game could be seen in the next figure.

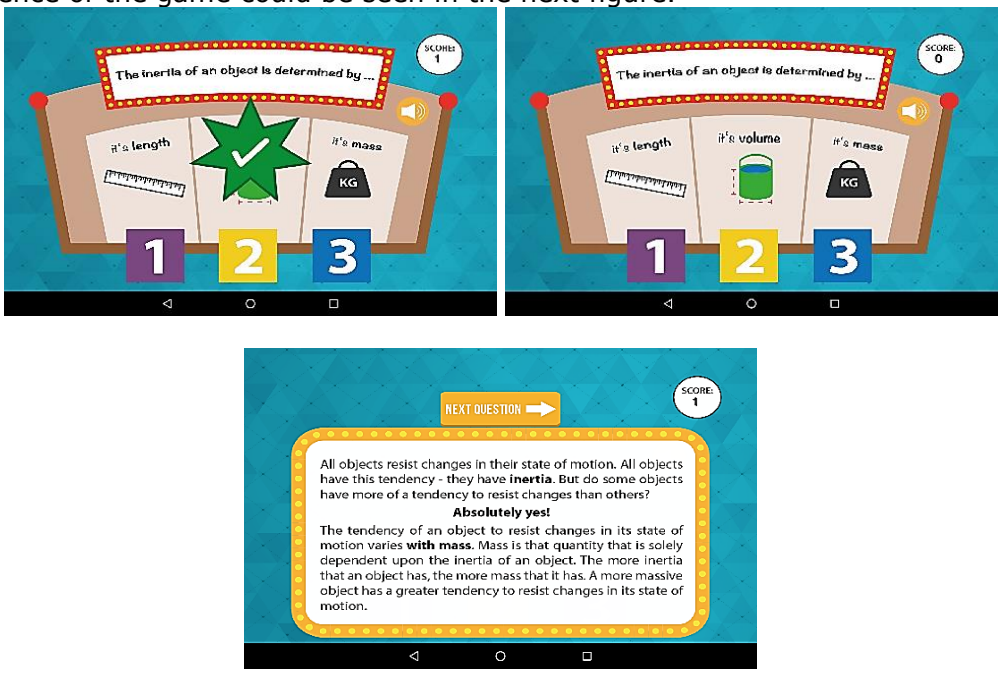


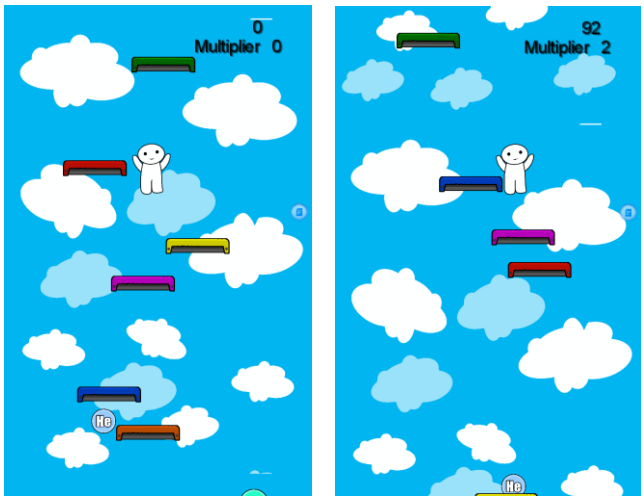
Figure 21: Game play: Quiz template

2.2.1.2 Second cycle of templates (Feburary 2017)

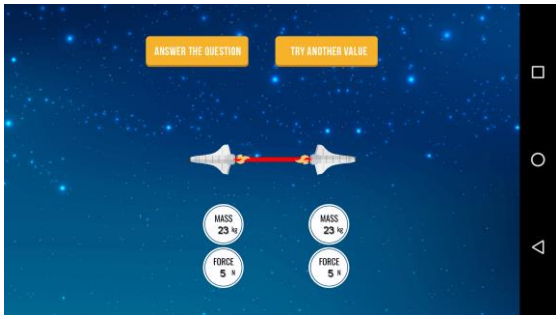
For the release in February 2017 five new game-modules have been developed, based on sub-genres of action, adventure, puzzle and quiz. These templates are described in the followed section:

Module 5: Action Platform - vertical scroller

Topic	Collect the correct "pick-ups" to get multiplier points, according to the asked question. Collecting the incorrect objects can also reduce the score. The character can move from platform to platform by jumping, to collect and avoid objects.
Game play	Jump on the platforms to get points. Catch the halogens to increase your points. Avoid the other symbols or you get minus points.
Enhancement by users	Add correct and incorrect objects. Define an overall question.
Learning goal	Learn/memorise the chemical symbols for the elements known as the halogens.

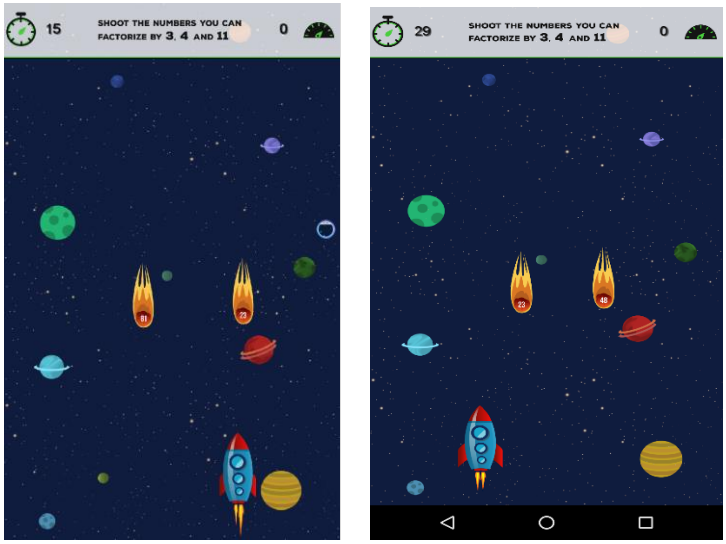
Preview	

Module 6: Phisical simulation

Topic	Physical experiments with mass, acceleration. Apply Newtons 2nd law of motion (F=m*a). Play around with the properties. Extra task: Simulation of Newtons 3rd law of motion (F1 = F2)
Game play	First level: See what happens if you change the force of the first rocket. The first rocket should reach the finish line at first. How to archive this. Calculate a=F/m Second level: Create an own rocket with a mass and acceleration. Define its force. Third level: See what happens if F1 < F2, F1 > F2 und F1 = F2. Answer the question which of Newtons laws is applied.
Enhancement by users	Change the force (by tapping of the rocket). Calculate the acceleration. Add an own object with mass and acceleration.
Learning goal	Learn about Newtons 2nd and 3rd law. Add an own object and program its behaviour.
Preview	

Module 7: Action Shooter

Topic	The player launches projectiles at asteroids with numbers.
Game play	Shoot at the asteroids who can be divided by 4, 3 and 11.
Enhancement by users	Define a new instruction e.g. numbers that could be divided by 5, 7, 12 and provide instruction on how to do this. Add looks with wrong and right numbers to the objects.
Learning goal	Maths: How to divide numbers correctly

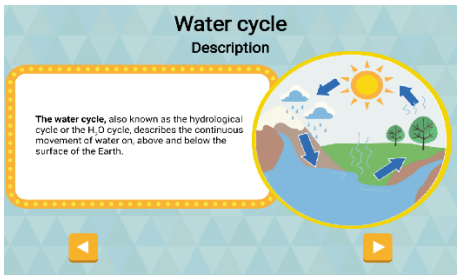
Preview	

Module 8: Lego NXT / Lego EV3

Topic	Control a NXT lego / EV3 robot and use therefore different sensors of your phone and of the robot.
Game play	Level one: Try to control the robot with the keys on the phone. Level two: Add code to control the robot with the inclination sensors of the phone Level three: Create a maze with the use of the Lego NXT / Lego EV3 button and ultrasonic sensor / or infrared sensor to detect obstacles
Enhancement by users	Level 2: Add code to define the range of the inclination sensors. The note bricks within the code helps the users. Level 3: Add code to define the range of the ultrasonic sensor to stop and move the motors of the robot.
Learning goal	Learn about sensor values, coordinates etc.
Preview	

Module 9: Adventure interactive book

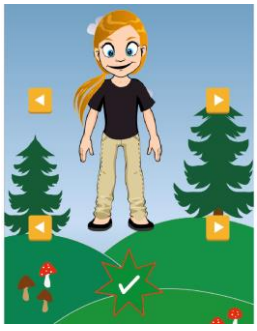
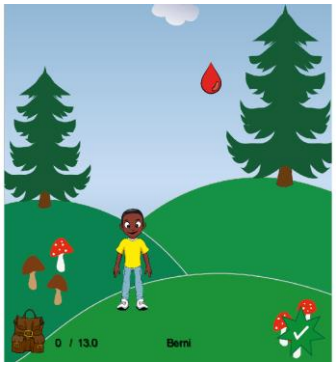
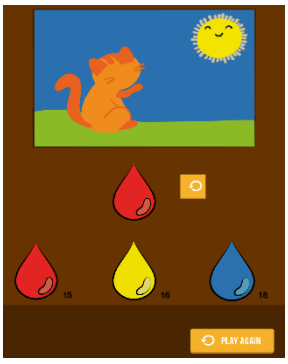
Topic	A very easy version of interactive book with scenes
Game play	Learn about different topics with the help of animations and descriptions. Create your own story.

Enhancement by users	Add texts and graphics/animations. Add more scenes and retell a story/topic.
Learning goal	Work with a certain topic
Preview	 

2.2.1.3 Third cycle of templates (May – June 2017)

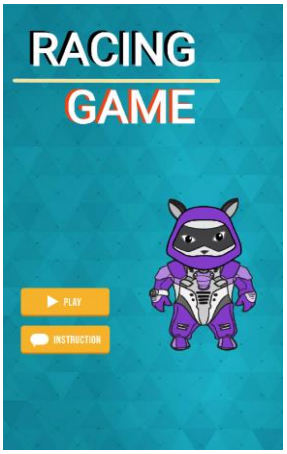
From May to June 2017 the last four templates have been developed and tested at our schools. These templates are described in the followed section:

Module 10: Adventure RPG

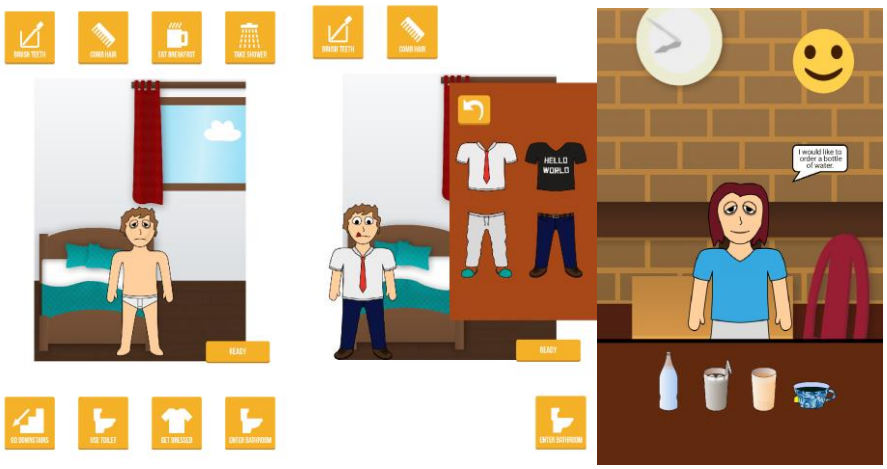
Topic	An adventure game where the user needs to collect inventory to build a project. In addition the user can create own characters.
Game play	Create your own character and choose between different players. Choose a project and collect items you need to build it. Collected items are in your inventory. You can drop them again. Finalize the project by mixing together the different parts.
Enhancement by users	Add your own project and items to collect.
Learning goal	Arts: Learn about the color theory by collecting colors and mixing them.
Preview	  

Module 11: Race simulation

Topic	One controls a transportation truck. The player has to avoid obstacles and collect trash.
Game play	Move the car by tilting your phone. Collect trash to get points and level up. Avoid the other cars to keep playing.

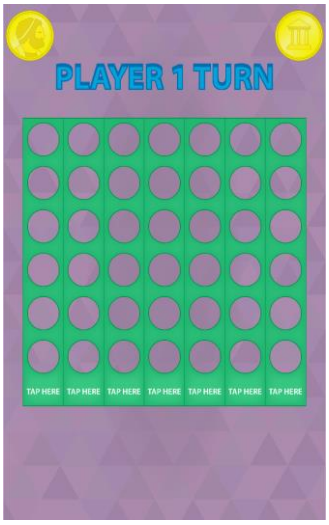
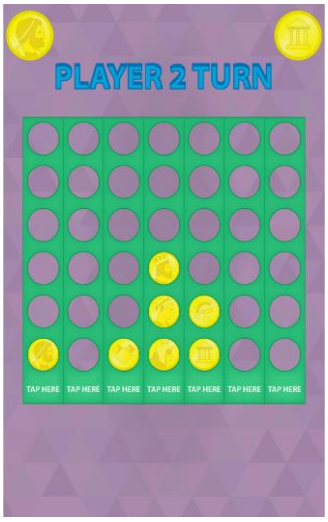
Enhancement by users	Add objects to increase the difficulty of the game and add more levels.
Learning goal	Natural Science: Save the city from pollution.
Preview	 

Modul 12: Life simulation

Topic	Cafe / life and work skills simulator for students with special needs - choosing the correct drinks order.
Game play	Help the character first to dress for work and after that at work: serve the right order of drings. Choose the right sequence for your morning routine and at work: Keep your customers happy by serving them the right drinks
Enhancement by users	Change the sequence according to your morning routing and at the cafè change the workplace - different items to order (change items, speech bubble and sound effects)
Learning goal	Change the variable sequence and adopt the game to your morning routine. To be able to identify equipment needed to complete a range of jobs. With some verbal prompting, correctly sequence some processes involved.
Preview	

Modul 13: Strategy

Topic	Multiplayer (on one tablet) Strategy decisions via a simple connect 4 template.

Game play	Have four coins in a row
Enhancement by users	Add pictures and code that fit to the subject
Learning goal	History: Learn about the different eares: greek, egypt, romans, kingdoms
Preview	 

2.2.2 Validation of templates

Assessment, feedback and validation is provided through the visualisation of analytic data in the Project Management Dashboard (PMD) including both qualitative and quantitative data. Through the integration of a BDSClientSDK (Big Data Services Client software development kit) in Create@School it is possible to explore information about users, their sessions, and actions, see Delivery 4.3. The PMD provides the framework needed for teachers to manage pupils and their projects. Explaining both assessment tools in more detail would be beyond the scope of this paper.

The analytics data that refers to the templates and results of the UX tests are presented within this section. This data was collected between February and April 2017. All in all 378 pupils (67 in Austria, 154 in Spain, and 157 in UK) took part in the project during this time period, see Table 3. The tracked data show that the event with the name "useTemplate" was logged 183 times. This means 183 games were created on the basis of a template. Further, the event with the name "createProgram" was logged 233 times in all pilot schools, which means these programs were created without the use of a template. The setup of the school units in which the templates has been used was very similar through all countries. Therefore the lesson plans' class teaching strategy was broken into three main sections: To start with (starter), Main Learning and Extension (plenary). The starter includes a debug or basic coding activity to support the learning, but it is the main learning section where pupils start coding, predominantly independently.

Country	Age	Pupils	Units	Template	Theme	Learning goal
UK	Yr 5 (9/10 year old)	30	3	Adventure	Science / Space	to apply their space themed written work in literacy to adventure games.

UK	Yr 8 (12/13 year old)	27	2	Quiz	Science / Energy	to demonstrate and assess their understanding of the unit of work as a whole.
UK	Yr 8 (12/13 year old)	27	2	Action Platform	Science	to collect halogens and avoid non-halogens.
Spain	Yr 8 (12/13 year old)	9	4	Interactive Book	Science / Biology	to create an interactive book on a vegetable garden.
Spain	Yr 9 (13/14 year old)	22	4	Interactive Book	Maths	to learn about logic and arithmetic.
Austria	Yr 12 (17/18 year old)	12	4	Quiz	Computer science	to create questions about the history of their school.
Austria	Yr 8 (12/13 year old)	25	4	Physical Simulation	Physics / Motion	to expand the template and add an own rocket.
Austria	Yr 9 (14/15 year old)	12	10	Adventure RPG	Computer science	to apply the template to one randomly allocated subject (e.g., biology or history

Table 2: Results of the experiments presented on countries, ages and themes

Based on the results from 2015, not only the program but also the whole packaging of our courses was adapted. For example, the pupils received a more general starter lesson about the app itself. Therefore, the template integrated the idea to let pupils first change the existing code, thus understanding the overall concept of the game template. In the second step, they had to add a similar object and apply the same concept they learnt from the previous level. Therefore, instead of four instruction units, in many classes only two were needed. The on-site observations showed that the classes who used a template understood how to apply formulas and concept better and solved the subject related problem. In contrast, classes who used no templates dealt more with programming/app problems. Thus the pupils who used a template felt more engaged, were more concentrated, and more of them reached the predefined learning goal as a result. It addition, it was seen as very positive that the template allowed personalization by adding pupil's own picture.

In general, all templates were seen as easy to understand, use and expand. For example, for the quiz template in both countries, pupils needed time to integrate the first question, but after they understood the overall concept, they added the other questions very quickly.
The challenge for developing the templates was, on the one hand, to pre-program the templates in an efficient way which provides pupils with functionalities to aid the process of building a new game; the templates must explain the important dynamics, mechanics, and aesthetics of a game (like coding how to reward the achievements or how to collect points). On the other hand, pupils should have the freedom to express themselves in a creative, fun, and dynamic way (e.g., to change images, sounds). Every class project should show them different methods to achieve the goals in a way which supports their logical thinking processes.

The results showed that the developed game templates encourage learning by doing, allow the expression of one's own ideas, and provide a visual programming language that is easy to understand and to learn. The project's goals were archived by linking the generated game templates to the program and design patterns of commercial games and thereby effectively support the development and adaptation of the learning material in a structured and replicable manner.

2.2.3 GPII Integration

The preliminary and testing integration of GPII in Create@School has been already described in Delivery 2.4.

The goal of integrating GPII into Create@School (Pocket Code) is to make the IDE more useable for various user groups. Software programs, mobile apps and websites have a default user interface that tries to cater for many people, but that is often unsuitable for people with special needs. Many such programs are adaptable, but on-site observations have shown that most people, never adapt the settings of the software they use. This may be because they think the default settings are all there is, because they are afraid of breaking something or because it is too difficult. Another reason is that devices at schools are not used every lesson by the same student.
Thus, from functional perspective Create@School allows the personalization of the game playing platform, through profiles the students can choose a preferred profile in Create@School whenever they want. The chosen profile only influences the Pocket Code app and not the general device settings.

Features integrated to adapt the UI for the users with special needs are: Choose Font face, Larger Text, High Contrast, Additional Icons, Large Icons, Large Spacing, Starter Bricks, Drag'n'Drop Delay, Icon Contrast and Hints and Tips.

The profile changing option has been embedded within the Create@School menu, and made profiles selectable under "a name" (to make them more distinctive and attractive for the user). We have named the profiles using names of gods to make profiles unique, attractive and to avoid discrimination. Each profile is linked to a special need usage. Therefore, Argus is a profile for visual (color) impairment, Odin for general visual impairment, Fenrir for motoric impairments and Tiro for a beginner mode. The following table shows all profiles, settings and their usage.

Usage	**Visual Impairment**		**Motoric Impairment**	**Beginner Mode**
	Visual (colour)	Visual (general)	Motoric	Learning
Special needs Profile name	**Argus**	**Odin**	**Fenrir**	**Tiro**
Large Text		X		
High Contrast	X	X		
Additional Icons	X			
Large Icons		X		
Large Spacing		X	X	
Starter Bricks				X
Drag'n'Drop Delay			X	

Table 3: Definition of features in special needs profiles

To use any kind GPII enabled preference setting or profiles the user interface of the application has to be adaptable respectively customizable. So the first step was to make the Create@School UI adaptable and customizable which means that most elements of UI had to be changeable at runtime by setting some

parameters. Below all the integrated functions for adapting Create@School are shown and described.

2.2.3.1 Choose the accessibility settings

To change any UI setting of Create@School and to use the GPII enabled features the accessibility settings are available within the normal settings menu of Create@School.

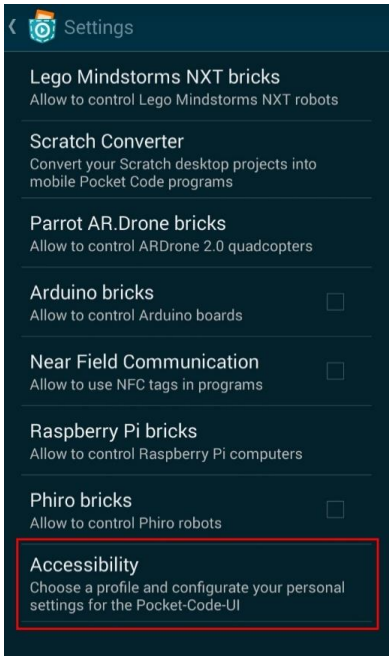


Figure 22: Accessibility Settings Menu

2.2.3.2 Accessibility Settings

In the accessibility menu the current active profile is visible and below all the integrated functions resp. settings which are available in Create@School are listed.
To change a setting, just tap onto one or more of the options. You will see that the *active profile* will change from Standard to *My Profile*. To apply your settings just press the *'Apply this profile'* button below. Remember, you always have to apply a profile before any changes happen.
You can choose a predefined GPII profile which is suited best to your needs. The predefined profile sets all settings needed, but allows you to activate or deactivate options, which do not feel well. To Select another predefined GPII profile press the *'Switch to predefined profiles'* button at the top.

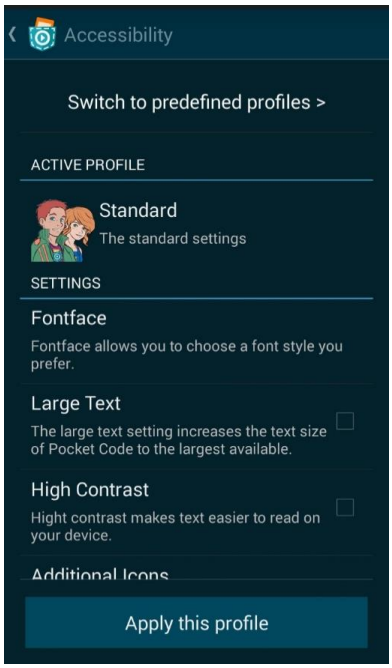


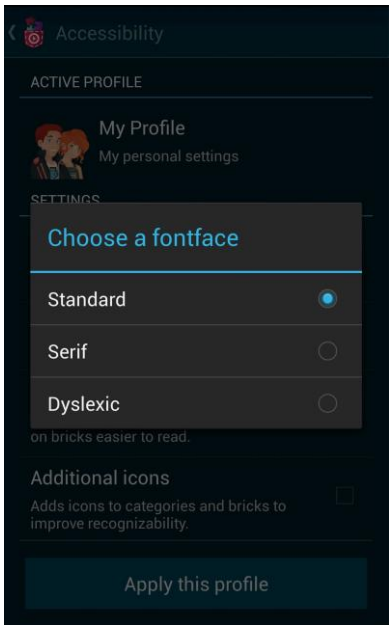
Figure 23: Accessibility Settings

2.2.3.3 Description of integrated accessibility features

Find below the description of features integrated in the beta version of the Create@School app.

Choose Font face

Choose Font face allows you to change the font face of the Create@School UI. The option *choose font face* let you choose between three different font faces. The Android *Standard*, *Serif* and *Dyslexic*. Serif changes the font face to a font with serifs. Dyslexic activates the font face OpenDyslexic, a font created to increase readability for readers with dyslexia. An example of the different font faces is given below.



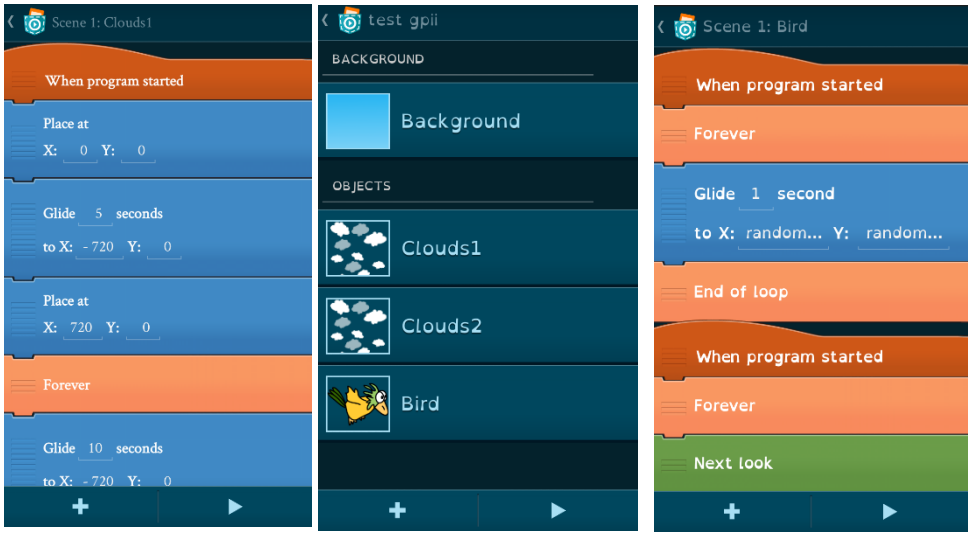
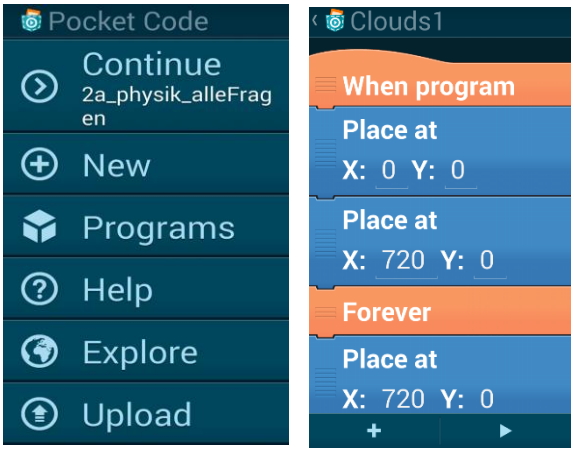


Figure 24: Fontface example

Choose Larger Text

The setting *Large Text* increases the font size of each element to improve readability. An example of views with large text is given below:



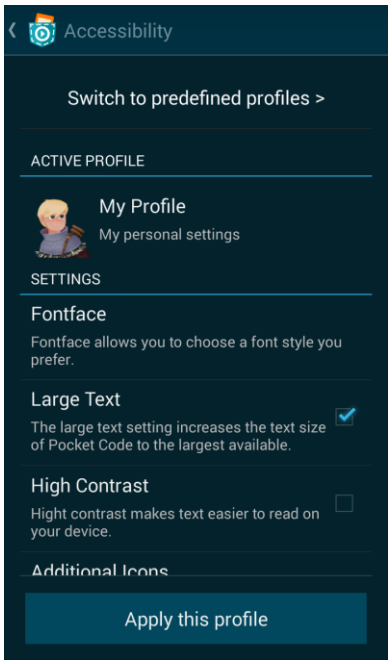
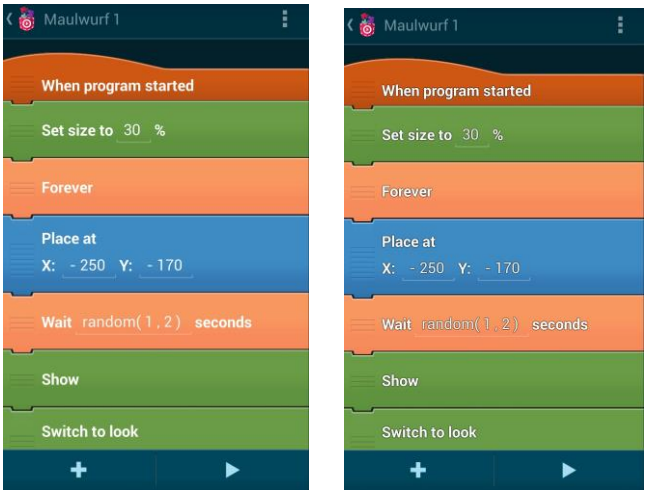


Figure 25: Larger Text example

High Contrast

The setting *High Contrast* increases the contrast of texts by adding an outline (shadow) to the text. A theming of the Create@School UI is in development. That means that in further versions it might be possible to change the background and text colour of every UI element which allows a better high contrast support. An example of views with high and low contrast is given below:



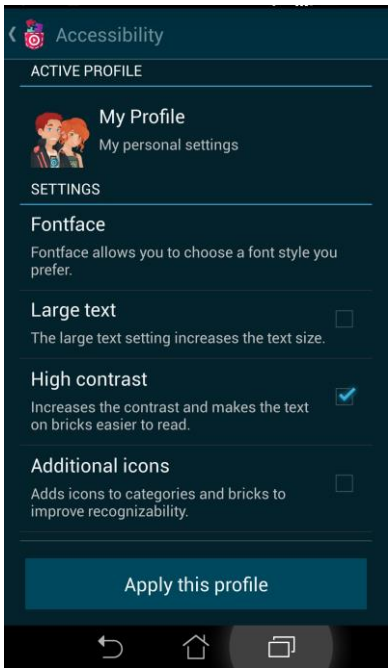
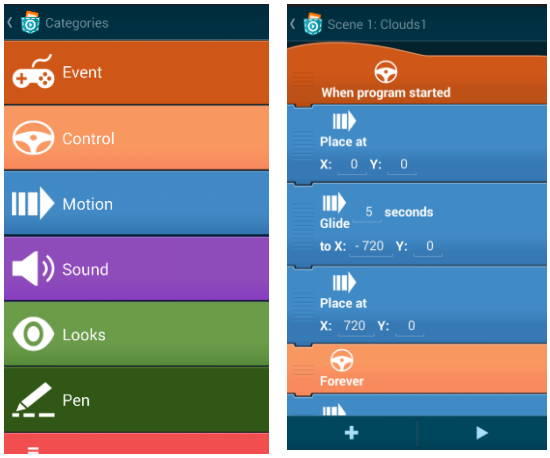


Figure 26: High Contrast example

Additional Icons

The setting *Additional Icons* provides additional icons for the brick categories and the bricks itself to avoid colour coded information. With additional icons it is possible to recognize the type of a brick even with any kind of colour vision impairment. Additional icons have been created for the categories *events, control, motion, sound, looks, pen* and *data.* An example of views with additional icons is given below:



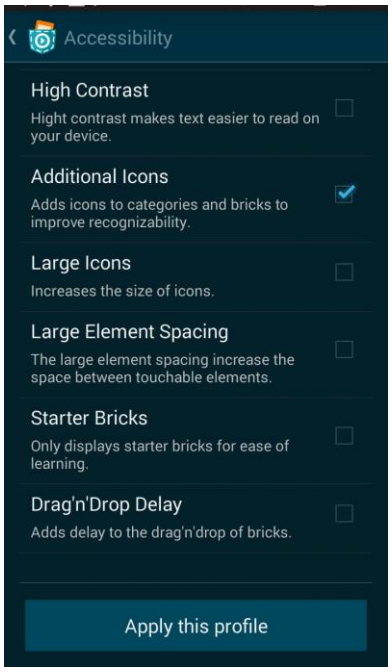
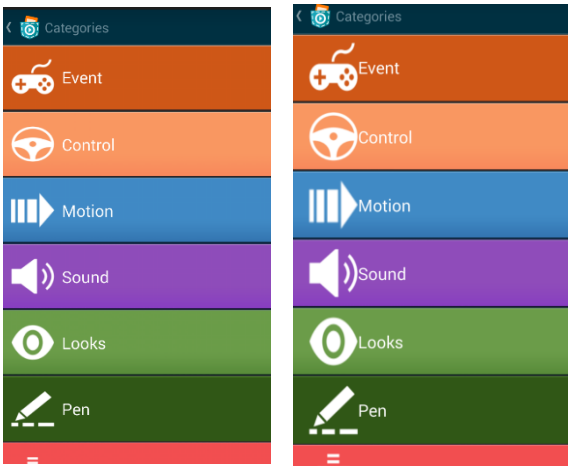


Figure 27: Additional Icons example

Large Icons

The setting Large *Icons* increases the size of the icons to improve the usability of Create@School for people with visual impairments. Moreover, the adaption and personalization of Create@School to the user's desires and the support for different devices can be increased. The setting affects most of the UI icons, additional icons included. An example of views with additional icons is given below:



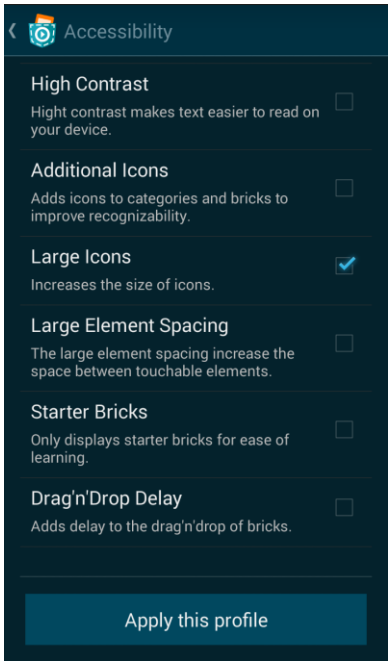
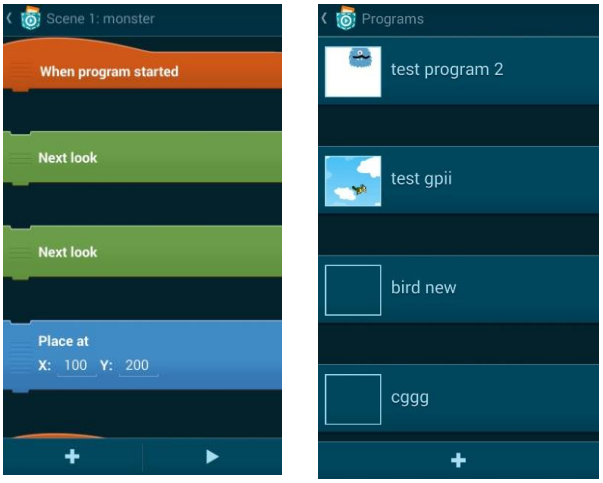


Figure 28: Large Icons example

Large Spacing

The setting *Large Element Spacing* increases the empty space between elements to get a clear separation and to make it easier to tap or click on elements and helps avoiding pressing the wrong button. The example below shows the program view and the bricks with large element spacing.



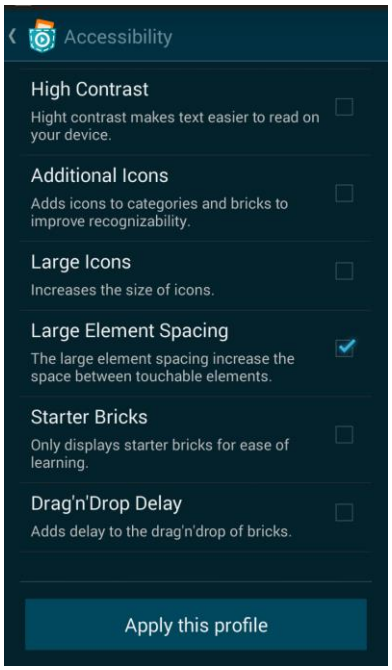
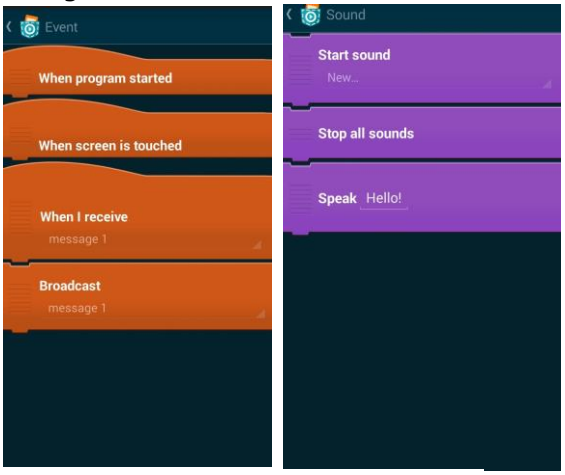


Figure 29: Large Spacing example

Starter Bricks

The setting *Starter Bricks* reduces the number of available bricks in every category to the most frequently used. This gives beginners a better overview of the options they have. An example of views is given below:



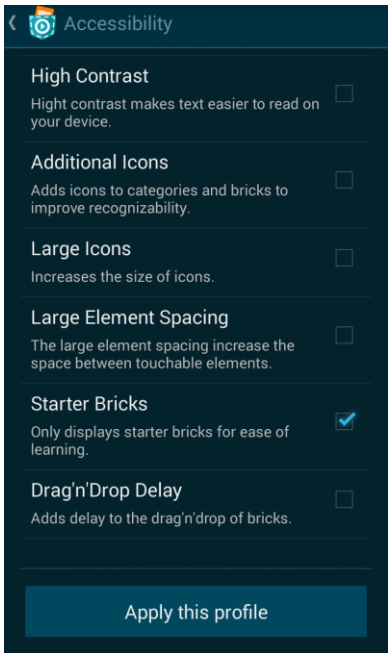
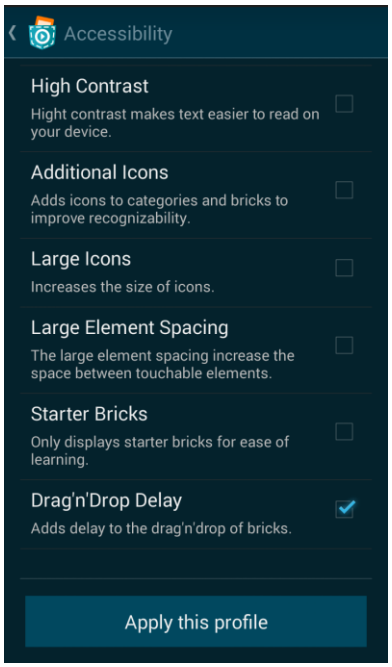


Figure 30: Starter Bricks example

Drag'n'Drop Delay

Figure 31: Drag'n'Drop example

The setting *Drag'n'Drop Delay* increases the time before a drag event happens by tapping on a brick (twice the regular time) to avoid wrong user input. The example below shows a drag event:



Icon Contrast

The setting *Icon Contrast* increases the contrast of the additional icons by inverting the graphic to improve the usability of Create@School for people with visual impairments. An example is given below:

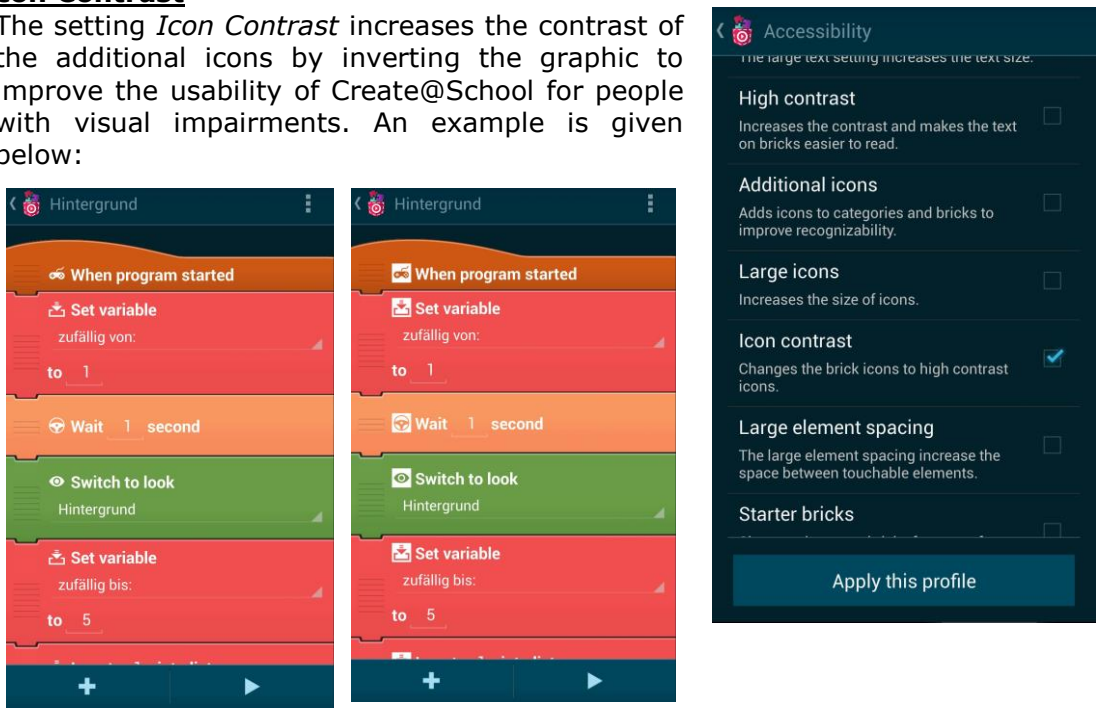
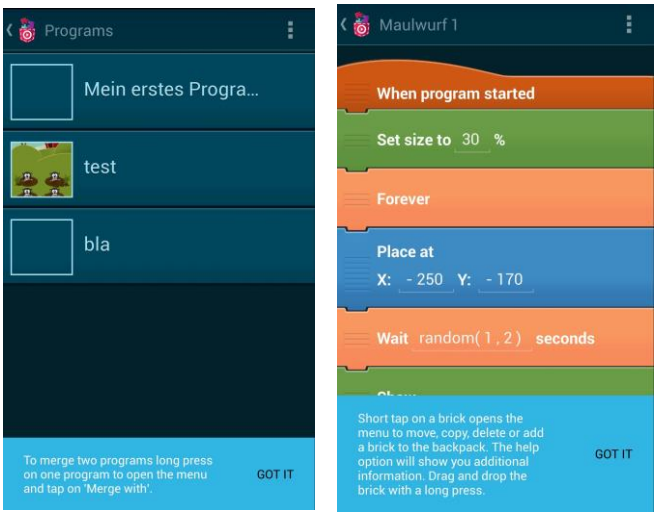


Figure 32: Icon Contrast example

Show Hints

By activating the *Show Hints* Create@School will give you hints at some points to let you what this view or option is all about. Hints are appearing at bottom of the screen. You can remove them and mark them as read by taping on 'GOT IT!' on the right. Show hints is active by default. An example of views with hints is given below:



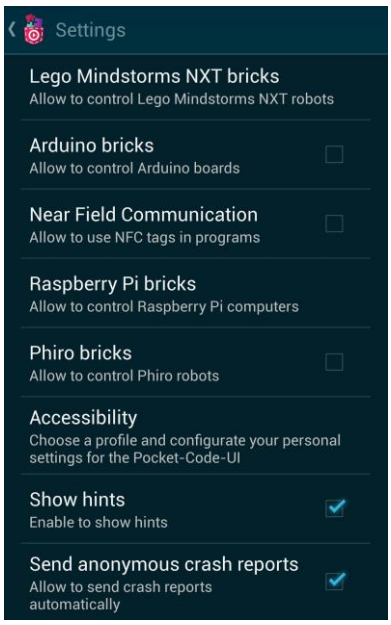


Figure 33: Show Hints example

2.2.3.4 Predefined Profiles

To adapt Create@School to different user groups we've specified five user groups, which includes a standard profile and four accessibility predefined profiles for special needs, with different properties. Every predefined profile activates or deactivates a set of the integrated functions which are described in chapter **¡Error! No se encuentra el origen de la referencia.**.
The provided profiles are available directly within the Create@School settings menu and selectable by name to make them more distinctive and attractive for the user. Especially for children the naming of the profiles can be very important to avoid any kind of discrimination. For this reason we use the names of gods to make them unique, attractive and to avoid discrimination. They are named Argus, Odin, Tiro and Fenrir. The fifth profile is the standard profile which resets the UI to the default settings.

In the predefined profiles menu you can see your current active profile at the top. Followed by the default '*Standard*' profile and your own customized profile called '*My Profile*'.
Furthermore there are four predefined profiles to adapt the Pocket Code UI to the user. They are called Argus, Odin, Fenrir and Tiro. Every one of them activates or deactivates a set of preferences from the accessibility settings menu.

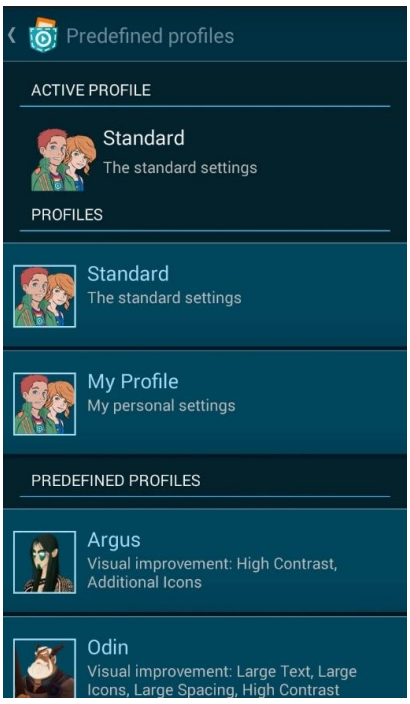


Figure 34: Predefined Profiles Menu

Every time you change any setting from the standard or from a predefined profile, your settings will be stored in the profile '*My Profile*'.

Figure 35: Predefined Profiles

Below there are the four predefined profiles to adapt the Create@School UI to the user. Every one of them activates or deactivates a set of preferences from the accessibility settings menu. A short description of the behaviour is shown below the profile name.

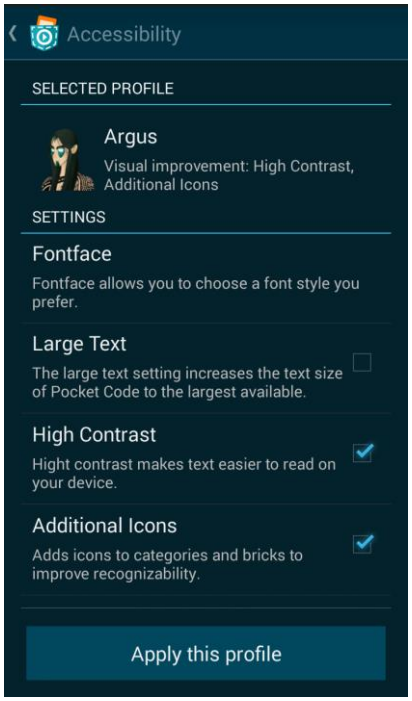
The table 7 showed already an overview of the settings and their influence on the UI.

Below the description of profiles integrated in the beta version of the Create@School App. The following screenshots shows the functional specifications of the new accessible profiles, however the graphic design might vary in the final version.

Profile Argus

The profile Argus activates the *high contrast* setting which improves the readability of the UI text. Furthermore, Argus activates the *additional icon* setting which adds additional icons to bricks and categories to improve the recognisability. The used icons have a higher contrast ratio than the default one.

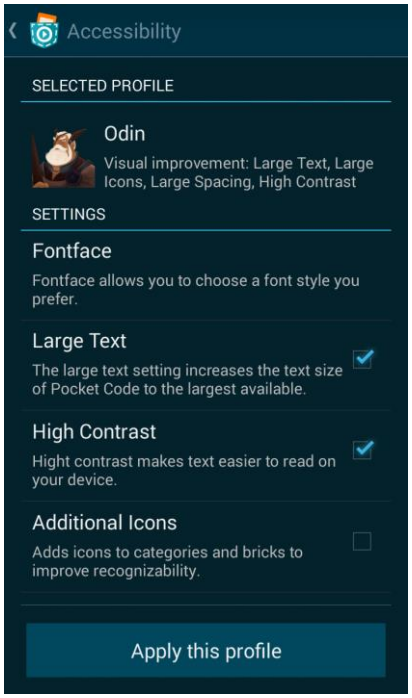
- High Contrast
- Additional Icons



Profile Odin

The profile Odin activates the settings *large text*, *large icons*, *large element spacing* and *high contrast.*
Large text increases the font size of elements and large icons enlarge all icons to improve the readability and remarkability. For this reason *large spacing* raises the space between elements to get a clear separation. Furthermore the contrast will be increased as by the Argus profile.

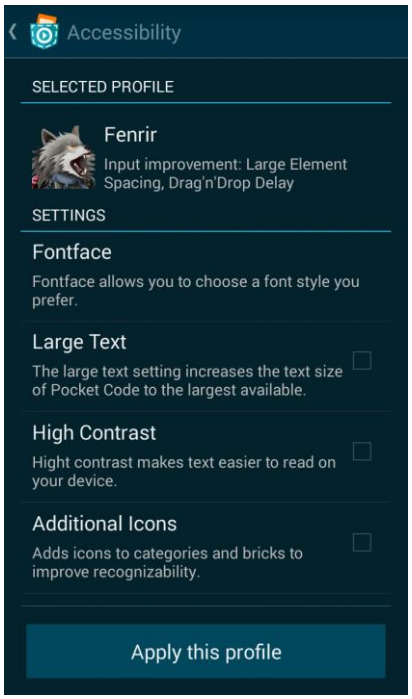
- Large Text
- Large Icons
- Large Element Spacing
- High Contrast



Profile Fenrir

The profile Fenrir activates the settings l*arge spacing* and *drag drop delay. Large element spacing* raises the space between elements to get a clear separation and to make it easier to tap on elements. For the same reason the drag'n'drop delay time of bricks will be increased to avoid wrong user input.

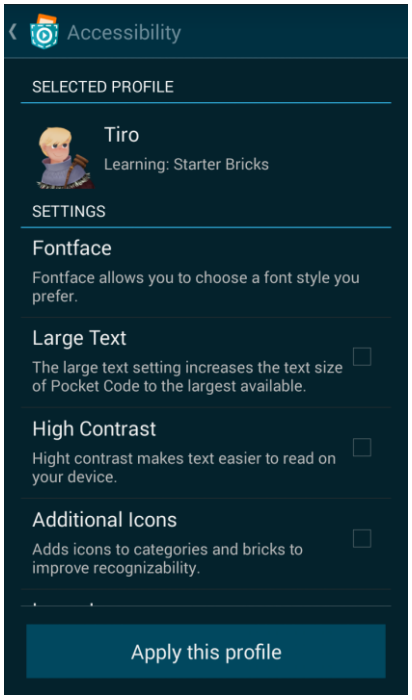
- Large Element Spacing
- Drag'n'Drop delay



Profile Tiro

The profile Tiro activates the setting *starter bricks*, which reduces the number of available bricks to the most frequently used. This gives beginners a better overview of the options they have.

- Starter Bricks



2.2.3.5 New GPII developments

The following developments/improvement have been integrated from March to June 2017.

Moving towards the automatic provision of individual settings from GPII / openAPE

At the moment Create@School has five (with the standard profile) profiles to personalize the UI. Individual settings are stored on the device and not on a GPII preference server. So if a pupil uses another device, his settings are not available on this one and everything must be customized manually again.
For this reason our next step was to develop GPII enabled featured to provide individual profiles which are stored in the cloud and are available on every device.

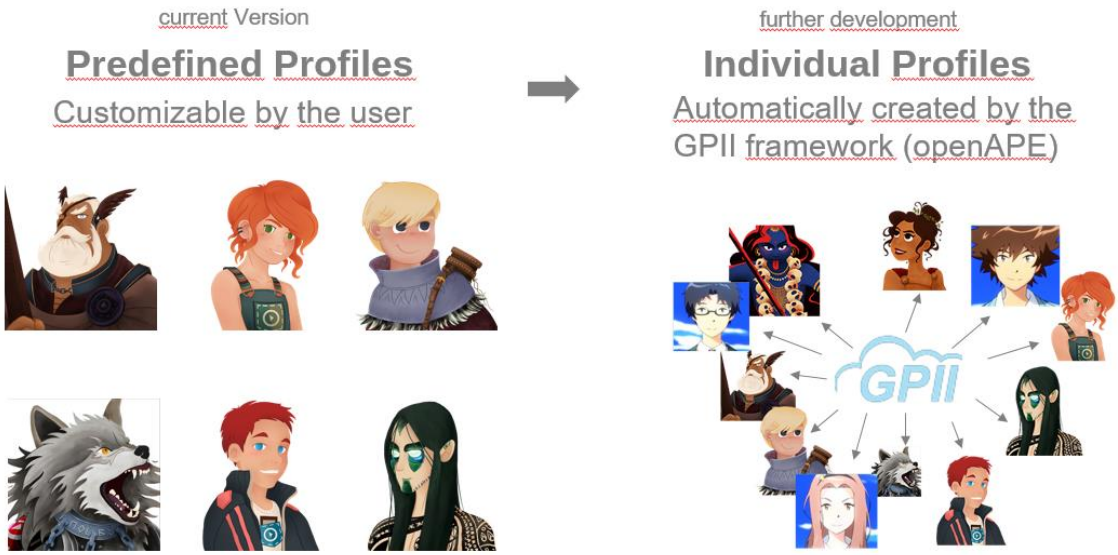


Figure 36: Predefined profiles vs. Individual profiles

The developed OpenAPE Framework provide GPII enabled features with individual profiles and settings to make Create@School more useable for various user groups. It is available at http://openape.gpii.eu/.

The developed OpenAPE infrastructure shown in Figure 37 below is based on the specifications defined in ISO/IEC 24752-8. This service provides an auto-adaptation for any application to the user's needs and desires. The main services are the following:

- Context services that can be used by any device to upload user preferences/settings, equipment, environment and task contexts (context of use) in order to make them globally available.
- Listing service that can be requested to get recommendations for UI settings and adaptations
- Resource service to make additional UI components available
- A feedback service to rate the proposed solution.

To provide individual profiles we additionally developed a token based RESTful user authentication system for OpenAPE. This allows users from Creat@School to log in to OpenAPE and customize their own profiles and to protect this data form unauthorized access.
Cause of system stability the OpenAPE features are not integrated in the current release of Create@School. The OpenAPE architecture is still experimental and - as a whole - not ready for production. However, we have integrated selected personalization features in the current release of Create@School that are proven and stable. This will ensure that Create@School will run stable as distributed to the masses, and that the OpenAPE architecture with its advanced features can be further scientifically evaluated beyond the project lifetime.

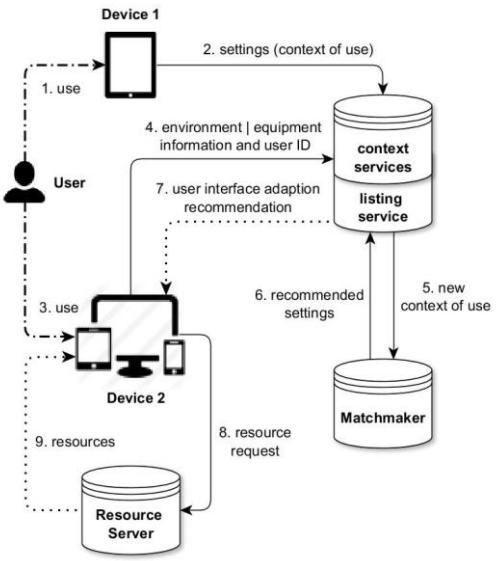


Figure 37: OpenAPE infrastructure

What the user sees and the openApe architecture is pictured in Figure 38.

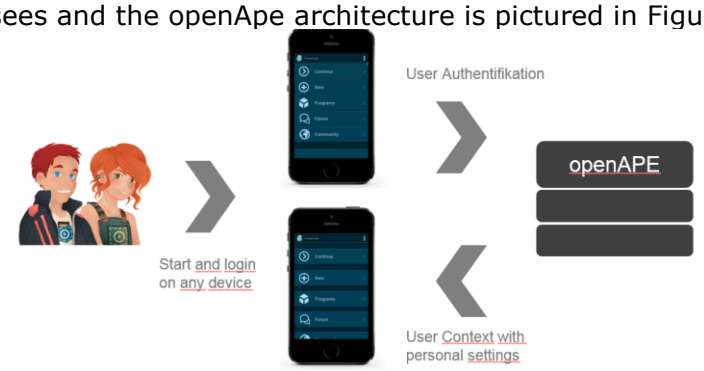


Figure 38: openAPE architecture

More adaptable settings

To make Create@School more adaptable and usable further settings have been defined at the beginning of the project. With a next release (planned for August 2017)

we integrated two more features: simple brick and an intro for the formula editor to support especially younger user, see description below:

- **Simple bricks**: Currently there are five bricks that are difficult to use for young children. Basically you can include also formulas in these bricks that are wrong or make no sense at all. Also Scratch simplifies these bricks by restrict them to certain entries. These modification affects the following bricks, a screenshot in Figure 39:
  - When 1<2 becomes true
  - If 1<2 is true then ...else....
  - if 1<2 is true then
  - What until 1<2 is true
  - Repeat until 1<2 is true

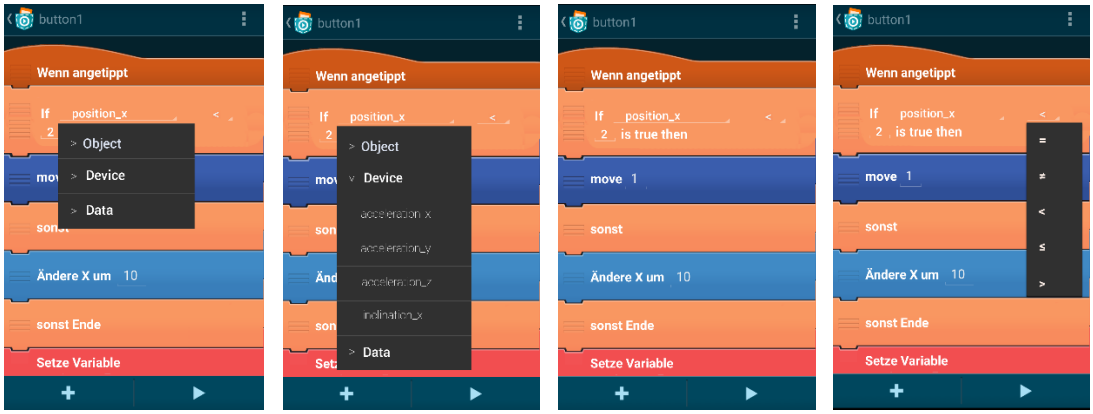


Figure 39: Screenshots of the simple bricks feature

- **Intro for Formula Editor**: Currently you get hints in many parts of the app e.g., when opening the script categories. For the Formula Editor we created more an intro than hints. The user can either skip the whole intro or click trough all steps, see Figure 40.

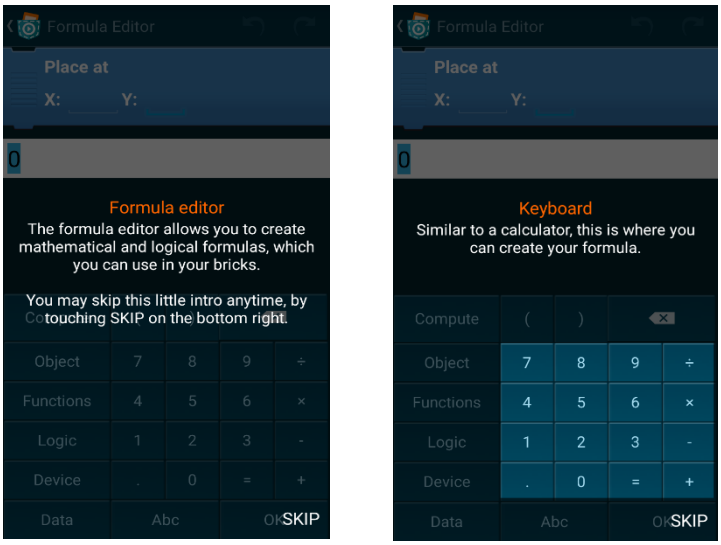


Figure 40: FE Intro

Some of the already implemented functions were be re-developed and improved. A list of all the relevant components is shown below:

- Apply the font face also for hints
- After "Apply Profile / Apply Settings" the user should always be in the main menu (it's not working e.g. on Moto G)
- Element spacing in objects makes problems when grouping the objects (a grey line occurs)
- The additional icons on bricks/the categories could be too small on different devices
- Apply font size in formula editor - should depend on the size of the device
- Font size and compute
- Font size and section in formula editor
- Big font size and numbers - cut the number or „…" (now it's an empty field on the bricks)
- Apply font face within the settings
- Apply contrast option to the hints
- Bug in element spacing: Some bricks are truncated
- add the "When tapped" brick to the starter bricks
- when using "Additional icons" and add a "Move 10 steps" brick: Pocket Code crashes

2.2.4 Enhancement of users' experience and usability

To achieve a long-term use in schools and to ensure the technology acceptance it is important to meet the user's requirements towards usability, ease of use, and satisfaction to foster the pupils' intent to use it.

For NOLB, the user experience (UE) comprised all aspect of the user's interaction with a product or service that influences the user's perceptions of the whole; this includes the layout, visual appearance, text, branding, sound and the interaction.

A study performed in the NOLB pilots investigated especially the user experience and usability of the app via a questionnaire survey. The aim of this survey was to find motivators and constraints for the usage of Pocket Code and to show significant differences among subgroups. The research showed required changes to make the app more accessible for different user subgroups in schools and to enable students to achieve their goals, and improve its ease of use. The outcomes influenced the development of the Create@School app, see Figure 41. The figure shows the version released in October 2016. In D1.2 two important features for redefining, creating and grouping objects and components were introduced: The scenes and grouping metaphor. With these two new features, it is now easier to maintain an overview in large projects and divide large projects in different smaller scenes.

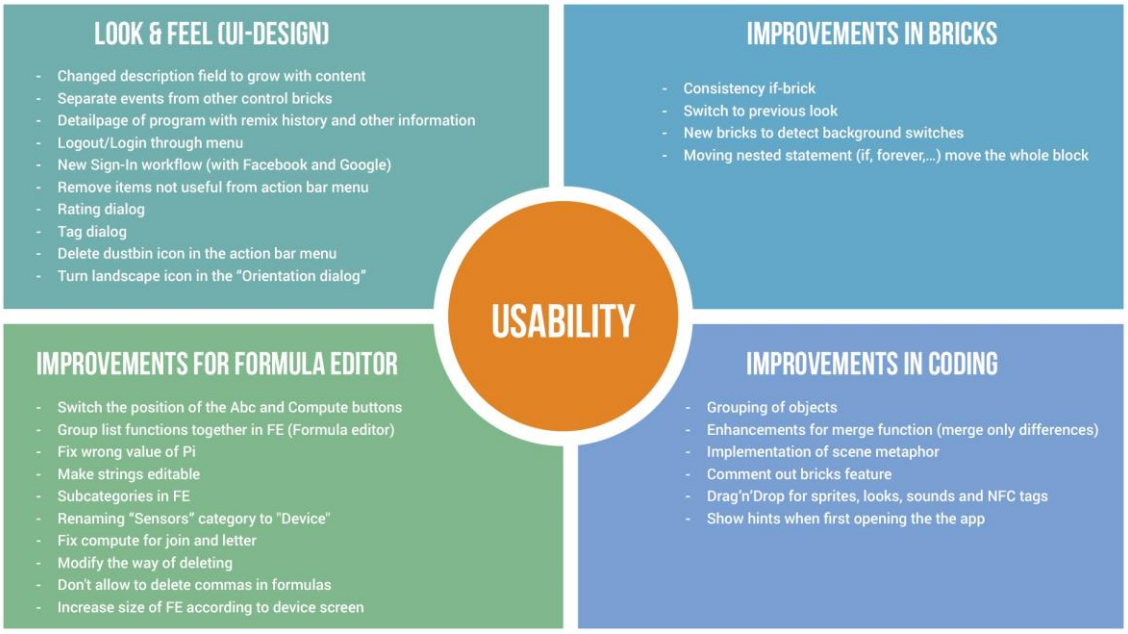


Figure 41: Improvements Create@School

In Febrary 2017 we had again a release of Create@School (version 5) with a number of usability improvements, see Table 4.

Task	Description
Improvements for merge function	• selecting itself should not be possible • merged programs: Data inconsistency in code.xml • fix bug that occurs when merging scenes
Renaming of strings	• rename brick 'When program started' to 'When program starts' • rename brick 'When I receive' to 'When you receive' • rename strings 'terms of use' to 'terms of use and service' • remove the colon after variable names • fix wrong radio button in overwrite-project dialog

Improvements for backpack	• fix hashcode of lookData and soundData • Fix backpacking/unpacking of variables and lists that are only appear in formulas
Formula editor	• "Compute" button: increase font size and contrast of displayed results • formula editor keys gradually disappear when formula gets longer
Improvements for scenes	• fixed scene copy • fix wrong scene reference • Fix wrong escaping of scene names (allow special characters) • Fix scene-reference in VariableActions (problems with local variables and scenes) • restart within scenes
Physics engine	• fixes selection of objects in "When physical collision" brick
Media library	• Media Library background URL based on screen orientation
Description field	• delete: show details in program overview (problems with scrolling bar) • create new fragment to show description of a project

Table 4: Usability improvements of version 5

In future the focus lies on a better first positive user experience and how to guide new programmers through the app.
The following points are therefore most desirable:

- in-app tutorials (e.g. FE intro)
- tutorial games / videos
- introduction when first starting the app (with the possibility to skip)
- more helpful hints (useful placed!)
- highlighting of new functionalities
- in-app banners to announce events e.g., game jams etc.
    - game jam support as system infrastructure to run game jams and to visualize the results of the jams.
- context sensitive help
- material design: more pictures instead of text, more direct navigation
- media packages
    - simple media package: enables to start your projects with decent graphics and sound
    - resources for different subjects

2.2.5 New functionalities to link with Scratch

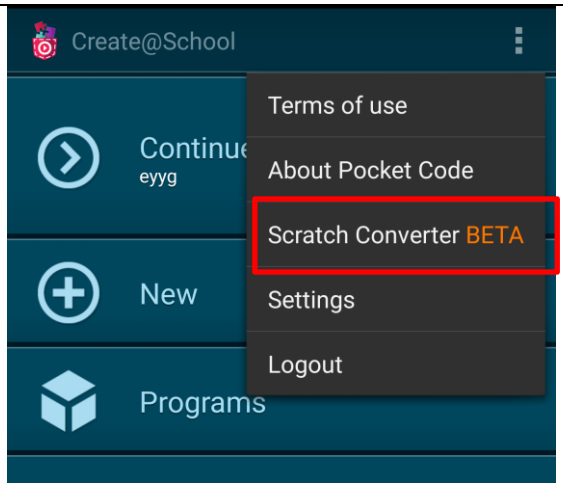
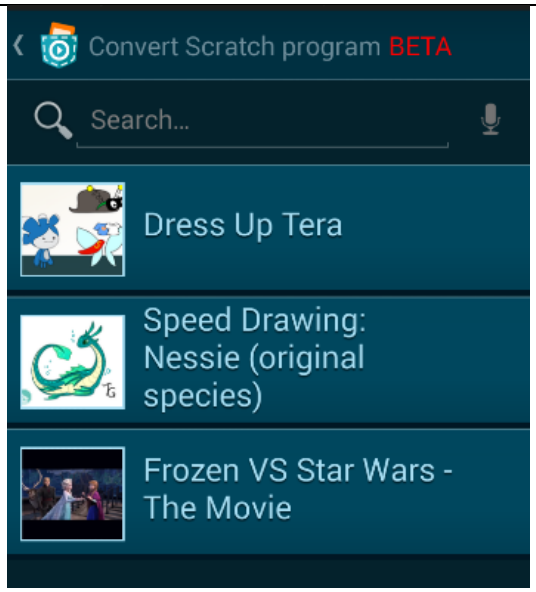
The development team not only worked in implemented the needed functionalities to become Scratch compliant and allow to use all Scratch bricks. This was especially important because a lot of teachers are already using Scratch at schools and Scratch is well-known among users all over the world [8]. As a result, Pocket Code released a Scratch Converter within its latest update to allow converting already written Scratch programs to the Catrobat language directly within the app, which also, most of them, have been integrated in Create@School.

This conversion project integrates a web-socket client (into the existing Pocket Code app) that communicates with the web service hosted on one of our servers. The Tornado web server (we are using) has been written by some former Google engineers and used by Facebook for some time. The web server uses certain asynchronous (non-blocking I/O) programming techniques in order to solve the C10k problem and keeps the memory footprint at a reasonably low level (vs. a multi-threaded approach as implemented by many other web servers like Apache). Thus, the web server should not be the underlying cause/bottleneck unless you receive more than thousands of requests per second.

The Scratch converter is also available through a web browser:

- http://scratch2.catrob.at/

How to use the Scratch Converter in Create@School is pictured in Table 3.

The converter is available through the Settings menu:	
Search field for finding Scratch programs and get suggestions	
Detail page of Scratch program with remix history and more information to the program and its author	

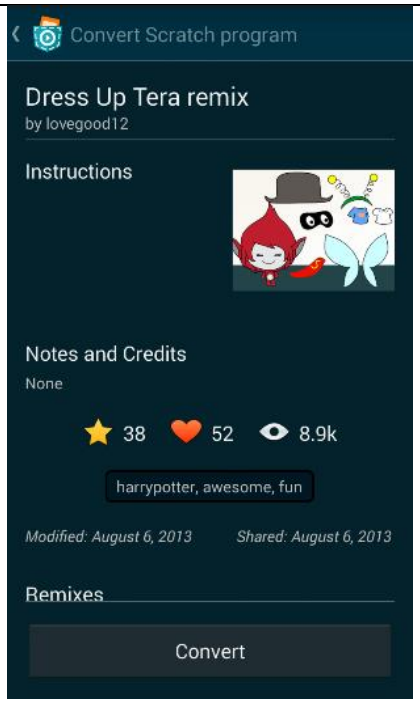
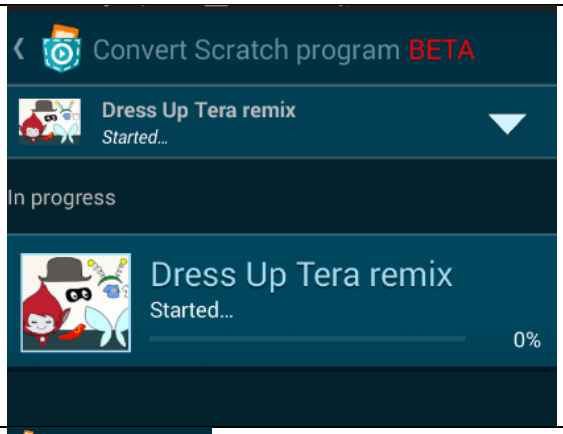
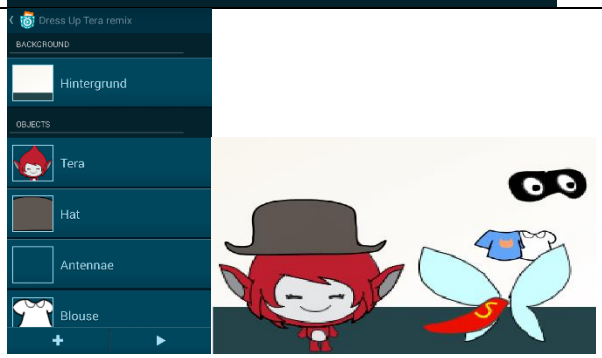
	
Conversion of the program in a swipe up panel	
Program opens in Create@School	

Table 5: Sequence for the scratch to Catrobat converter

For Create@School almost all Scratch bricks have been also integrated into Create@School. Figure 42 shows the latest update of Scratch bricks:

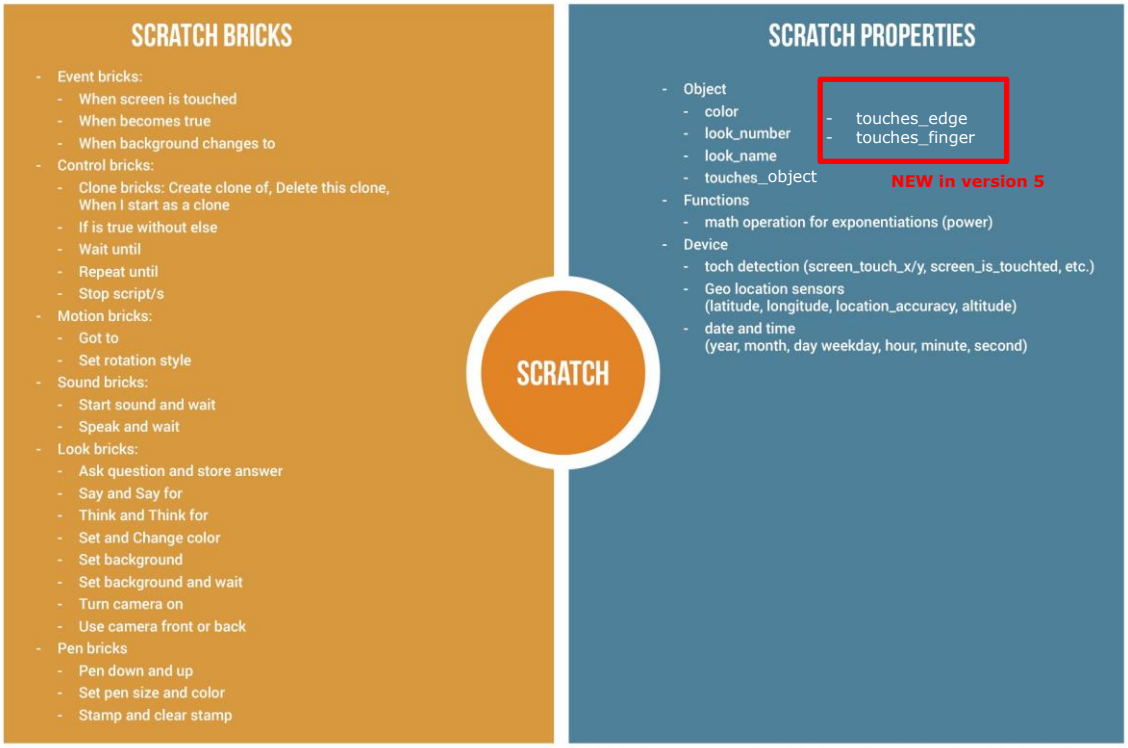


Figure 42: Implemented Scratch functionalities

The implemented Scratch Converter is still a beta version because we are still not Scratch complete. See the overview of still missing Scratch functionalities in the next figure:

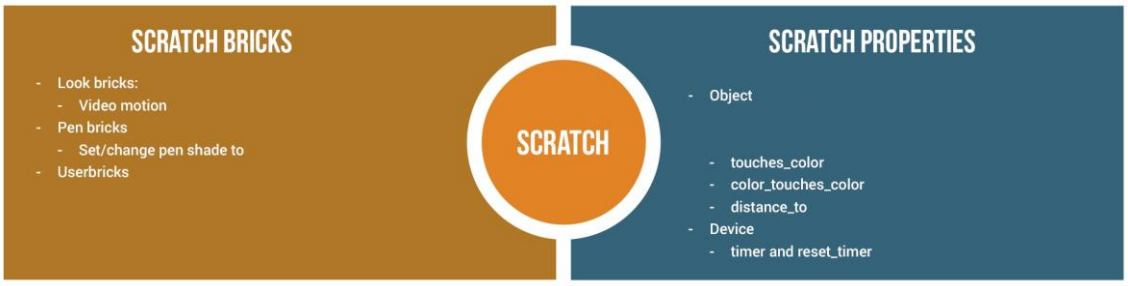


Figure 43: Missing Scratch functionalities June 2017

In addition two more bricks have been implemented (which were not at any of the lists):

- New brick in sound category for recording answers and store them in a variable (version 5)
- New brick in look category to switch the look by number (version 8)

2.2.6 New hardware extensions

In order to provide interactive classroom gamification with other educational hardware extensions, we considered how to integrate existing educational hardware (Arduino, Raspberry Pi, LEGO NXT,…) into Create@School. This should help to facilitate the large scale adoption of Create@School and NOLB's teaching framework as part of the roll out strategy. In this regard during the last year the following extensions were integrated:

2.2.6.1 Physics Engine

This enables the user to define certain physical features to objects and the stage e.g. the definition of velocity, gravity, mass, bounce factor or friction, to create simple simulations of real world experiments. An overview about all the bricks is shown in figure 40.

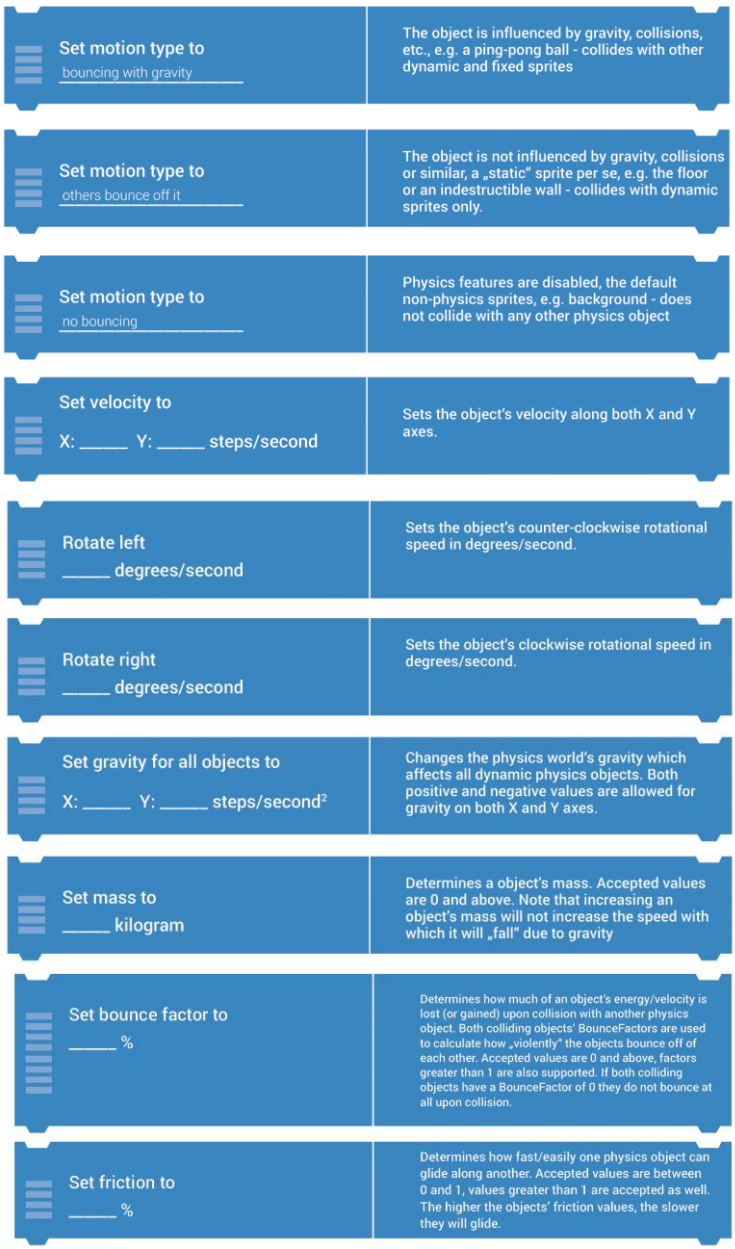


Figure 44: Physics engine extensions

2.2.6.2 Near field communication (NFC) tags

These allow the user to execute certain parts of the code when a specific or any NFC tag is triggered. NFC combines the functionality of the CARD interface and the functionality of the READER / WRITER interface in one chip. The integrated software stack allows to use the chip in a mobile device in both operation modes (card / reader) automatically. Figure 41 shows the connection from RFID and NFC.

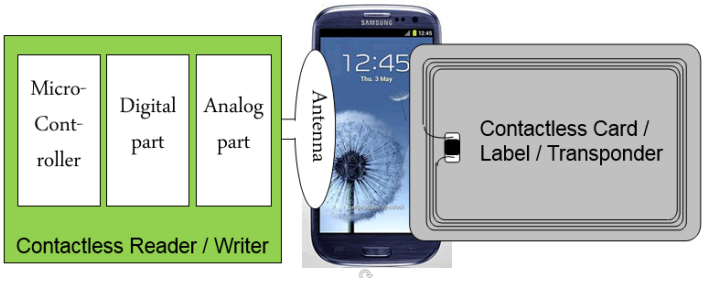


Figure 45: From RFID to NFC

2.2.6.3 Arduino

You are able to control an Arduino board with the Arduino Standard Firmata Library via Bluetooth with the Pocket Code application. The user is able to change the Digital and PWM (Pulse Width Modulation) Pins and read the Analog Pins of the board. You can create your own Arduino project in just a few minutes by using the Arduino bricks and sensor functions and implement your Arduino board actively into provided, or new Pocket Code applications. Figure 42 shows a program that creates a disco light with by setting led pins to different colours.

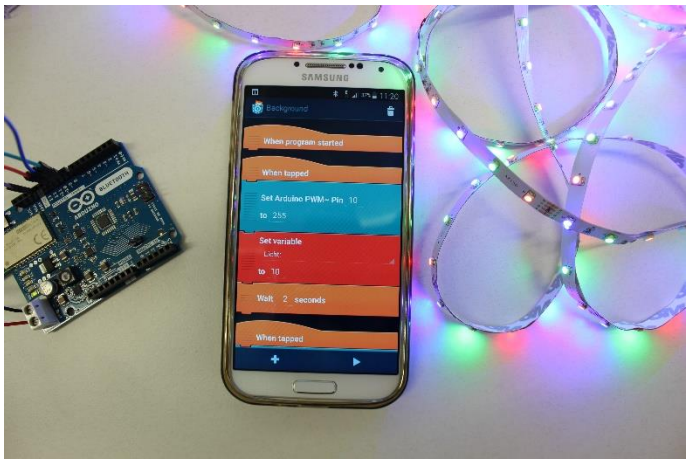


Figure 46: Control an Arduino board with Create@School

2.2.6.4 Raspberry Pi

The aim of this sub-project was to allow Pocket Code to control a Raspberry Pi remotely in order to enable children and novices to acquire knowledge and arouse interest in both programming and electronics. Features of the Raspberry Pi can be accessed without having to write code on a Raspberry Pi - all important features are accessible via special bricks within Create@School.

There are lots of possible applications, e.g., home-automation, robots or mood lights (IoT).

2.2.6.5 Phiro robot

Phiro robot from India: http://www.robotixedu.com/ we used in schools during the project. See the bricks in Figure 47.

Figure 47: Phiro bricks for motor, sound, and lights

2.2.6.6 EV3 lego robot

The EV3 Lego robot has been integrated in version 5 of Create@School, see the bricks in Figure 48.

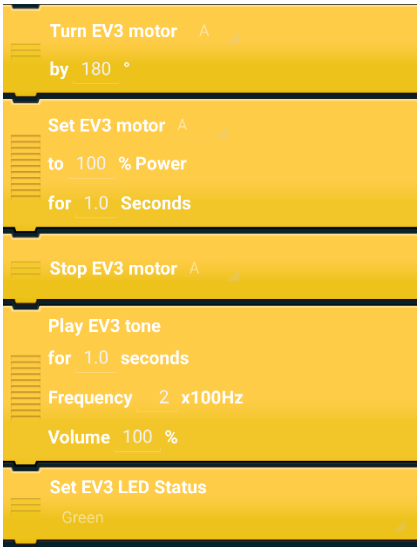


Figure 48: EV3 Lego robot - new bricks

2.2.6.7 Chromecast

Adding Google Cast support to Catroid. With this version of Create@School students are able to cast the stage to a cast device like Chromecast, see Figure 48 To control the stage a simple gamepad is used at the moment.

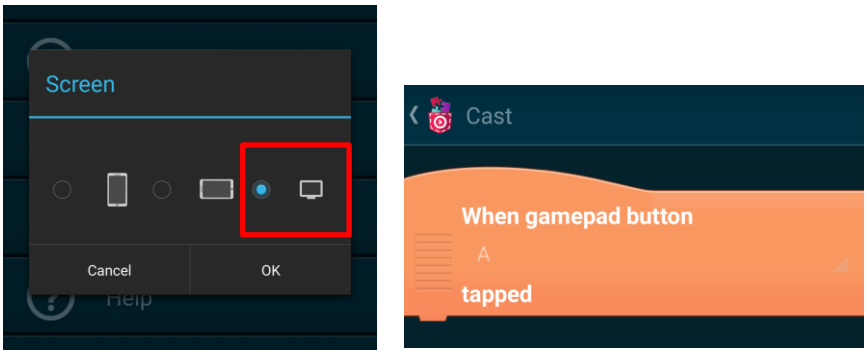


Figure 49: Chromecast feature

2.2.6.8 Extensions in development

The following extensions are still in development:

- AR Drone and Jumping Sumo drone: For controlling the drone provided from Parrot.
- LittleBits: Easy to use electronic building blocks

2.2.7 Support of new operating systems/environments

- D4 OS Version (Catty): Catty, also known as Pocket Code for iOS, is an on-device visual programming system for iPhones. A closed beta version is already available for registered users.
- HTML 5: The HTML 5 player is a web-based player using PHP (restful services) at the sever and JavaScript on the client side to play Pocket Code programs directly in the browser without downloading the program first. Figure 5 displays the Pythagorean theorem- Game which could be played within the browser.

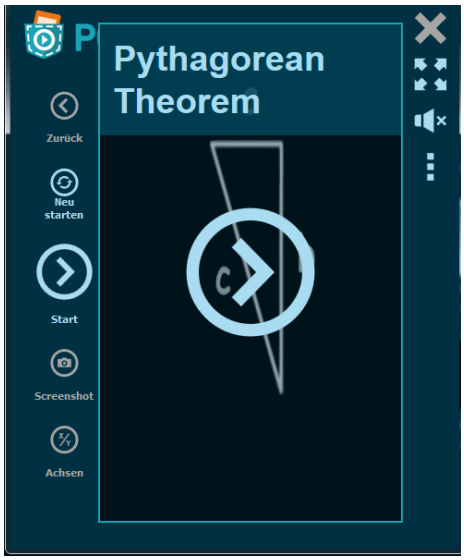


Figure 50: HTML5 player on the Catrobat web share

2.2.8 Like and remixing features on the Web-View

In March 2017 a number of new features within our web-share has been released (https://share.catrob.at/pocketcode/). Figure 51 shows screenshots to all features.

Like-buttons: Every program can now be rated with different sybols. Basically, a like rating system is a popularity measuring system that offers only a single option to the user, i.e. to like an item. Such like-only rating systems are quite common on social networks, where dislike ratings could have a negative impact on the overall user satisfaction. The like buttons are shown on the detail page of every uploaded program. We use four different emoticon buttons, including "thumbs up", "laugh", "love", and "wow".

See similar programs and get recommendations:

- Option: "User who downloaded this program also downloaded"

Like on Amazon a user see programs that are similar to his own program. The similarity is determindend e.g., by the use of same tags.

- Option: Recommended programs section

On the main page we implemented a new section "Recommended programs". The more likes a program receives, the higher it appears in this list.

Remix graph: On every detail page of a program a new button appear: "Show remix graph". This allows user to see the remix graph of their program. The nodes (i.e., programs) are arranged in chronological order from top to bottom. Each program to which an arrow points (i.e., incoming link or incoming directed edge) is a child program (i.e., remix) of another program (i.e., its parent program). A remix program is a program that has been downloaded by another user and uploaded again with changes (e.g., new scripts or images has been added).

Notifications: Everytime a program gets remixed the user who crated the origin program get a notification.

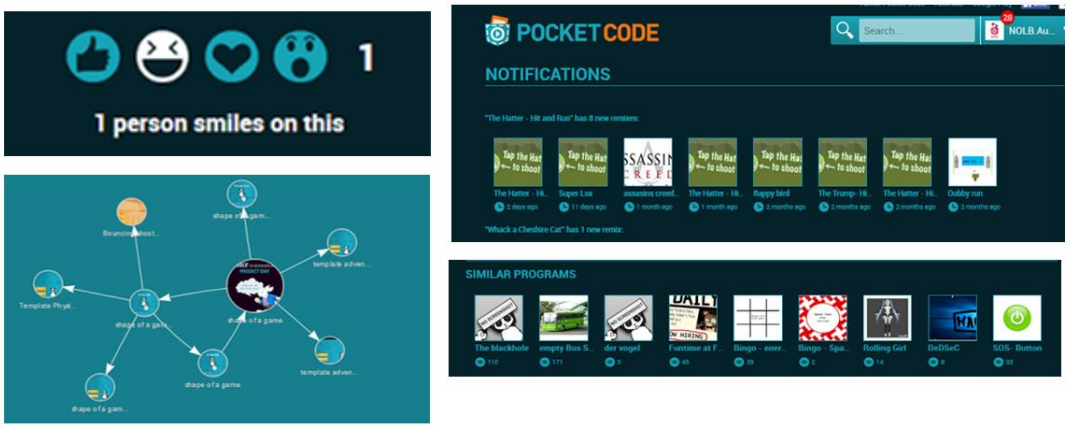


Figure 51: new remix and like features

3 THE PROJECT MANAGEMENT DASHBOARD (PMD) AND ANALITYCS TOOL

The PMD and analytical tool are web services that allow the follow up process in classes, generate feedback and improve teaching skills, capacities, planning and processes.

3.1 Project Management Dashboard (PMD)

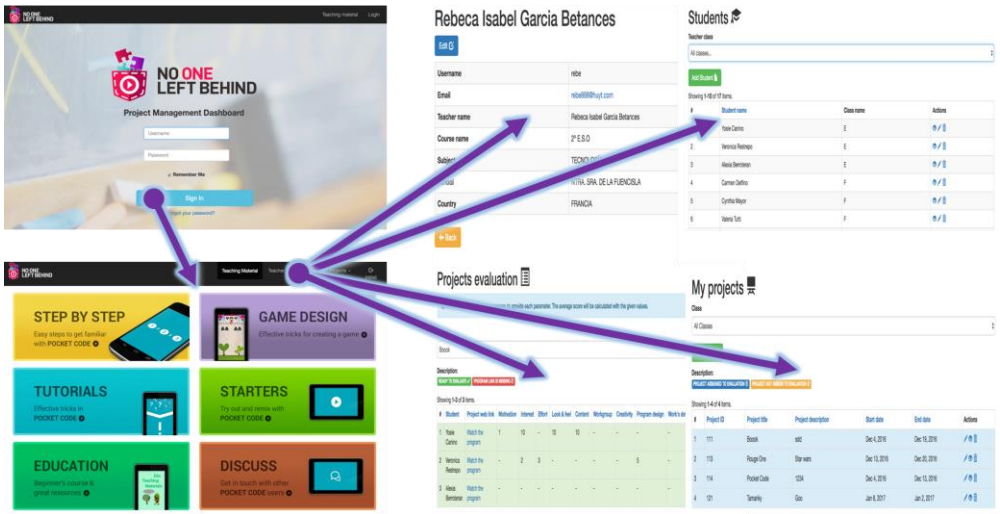


Figure 52: Overveiw PMD

The PMD is a web interface that allows orchestration of the learning environment and enables integration and engagement of all students in class. Through the PMD the teachers can plan, assign and manage the delivery of game projects to support new game based teaching approaches. Also, the PMD allows qualitative evaluation of students regarding the completion of projects and achievements of objectives.

The PMD allows to manage students and classes for each teacher of No One Left Behind project. The entry point of the PMD is the log in page, after the login the teacher has access to a set of tabs that offer several functionalities that are related to Create@School and the management of students, classes, project and their evaluation. The PMD is accessible through this link: https://www.pmdnolb.cloud/

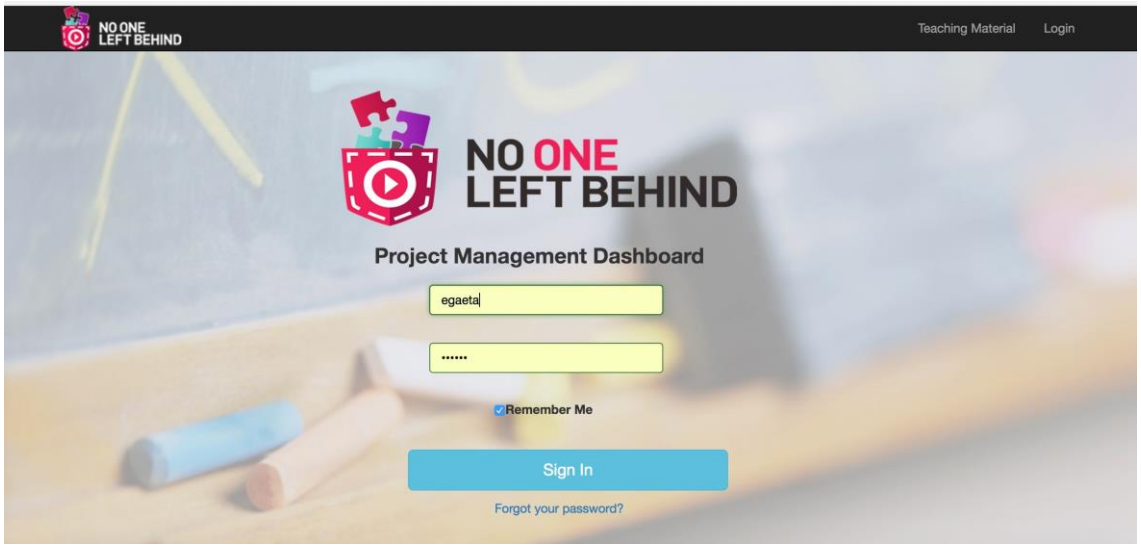


Figure 53: PMD Login

3.1.1 Teaching material tab

The first tab that appears after the login is the "Teaching Material" tab. Within this tab the teacher has a collection of learning resources about Create@School.

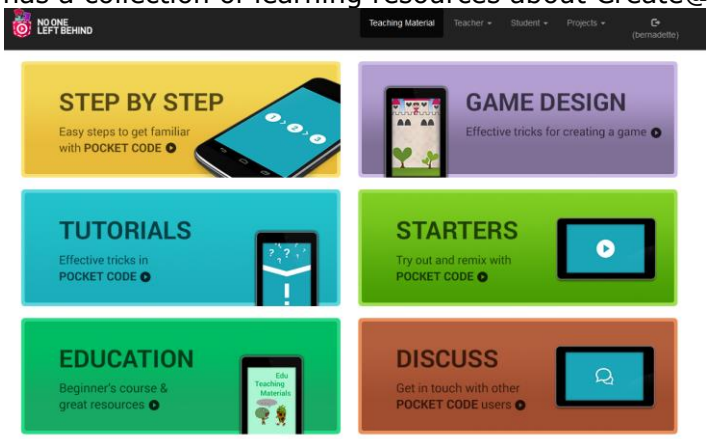


Figure 54: Teaching material tab

3.1.2 Teacher tab

The second tab, that appears after the login is the "Teacher" tab. Within this tab the teacher can configure his personal information and set his classes relationships.
The class concepts, at this point are just relationships. Inside the following tab "Student", the teacher can create and associate a set of students to the classes he or she has defined within this tab.

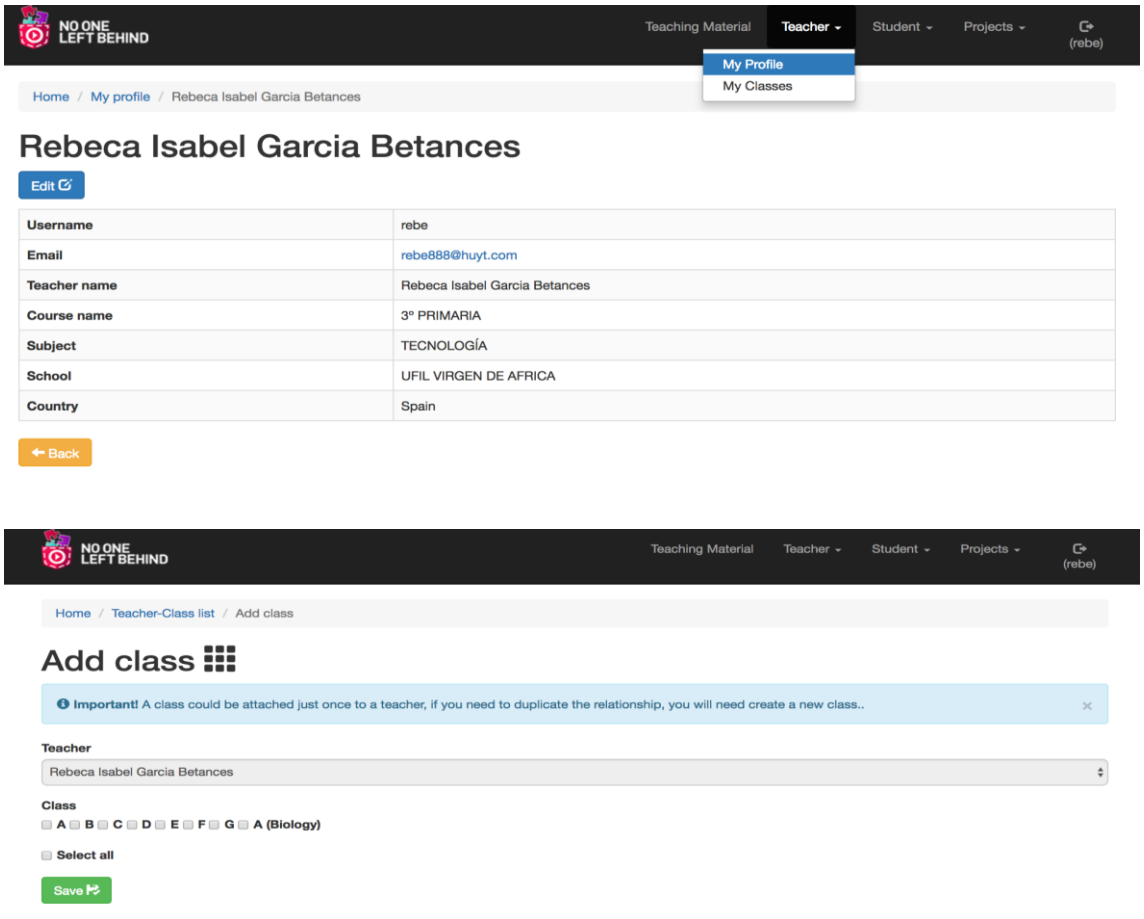


Figure 55: Teacher tab

3.1.3 Student tab

The "Student tab" allows the teachers to add new students or to edit them if already exist and also to associate the students to their classes.

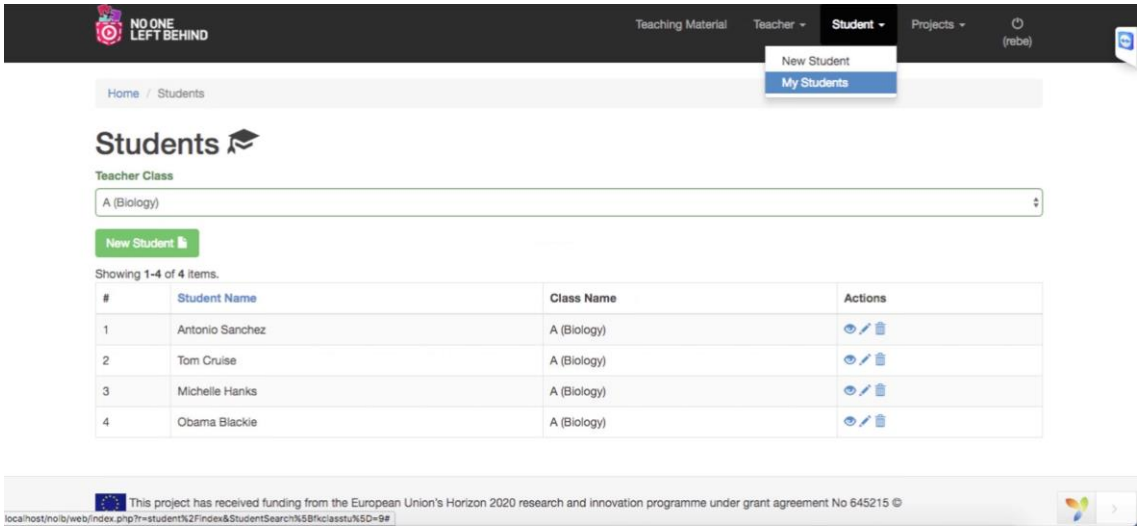


Figure 56: Students' tab

For each student, that the teacher adds, he or she has also to set the Create@School username. In this way, later when the student uploads a work with Create@School, the teacher can see this work inside the PMD.

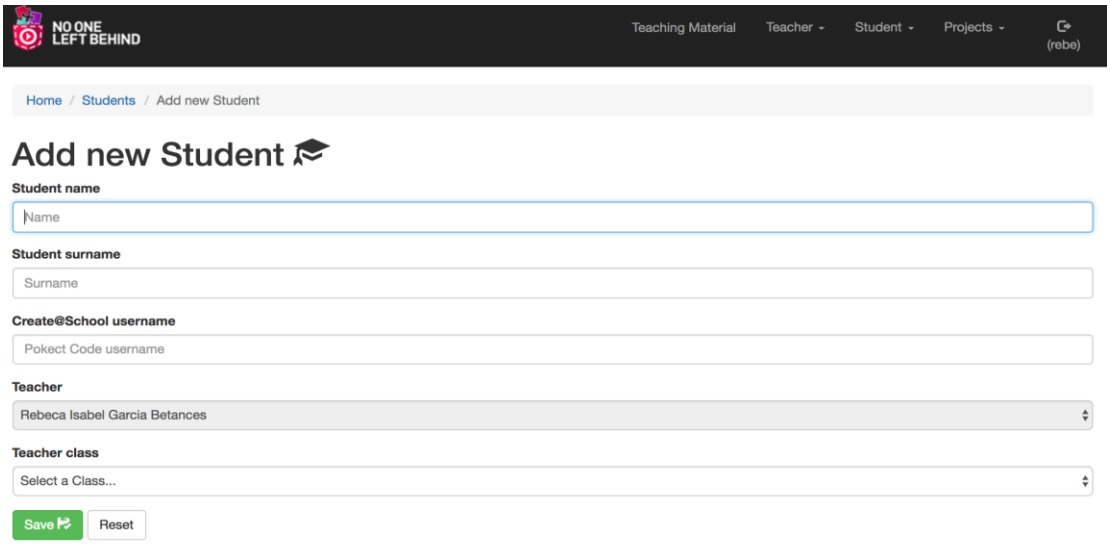


Figure 57: Student tab - add student

3.1.4 Projects tab

The "Projects tab" allows the teachers to add projects for the students of their classes.

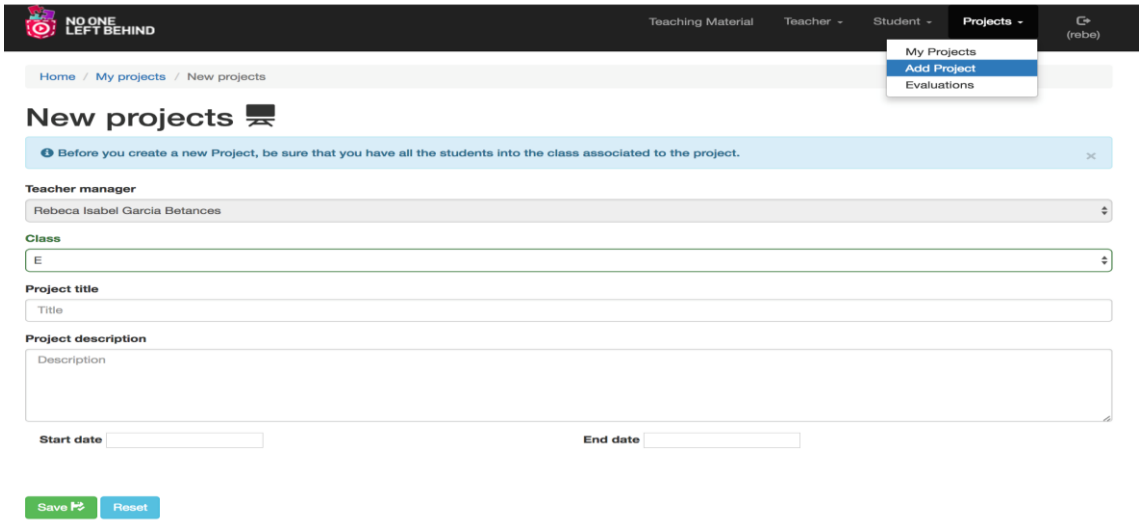


Figure 58: Projects tab – add projects

After the creation of a project the teacher can open the evaluation tab for this project. When the evaluation tab is open the teacher is able to review the classwork updated by the students and can evaluate each one by assigning values to a set of parameters that he or she considers relevant for the evaluation of the student.

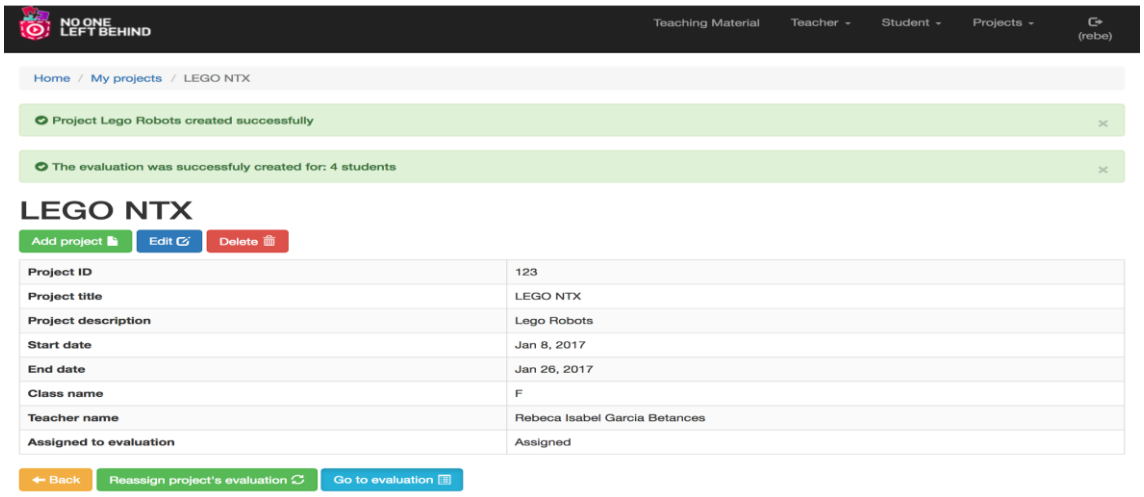


Figure 59: Projects tab – Projects view after adding a project

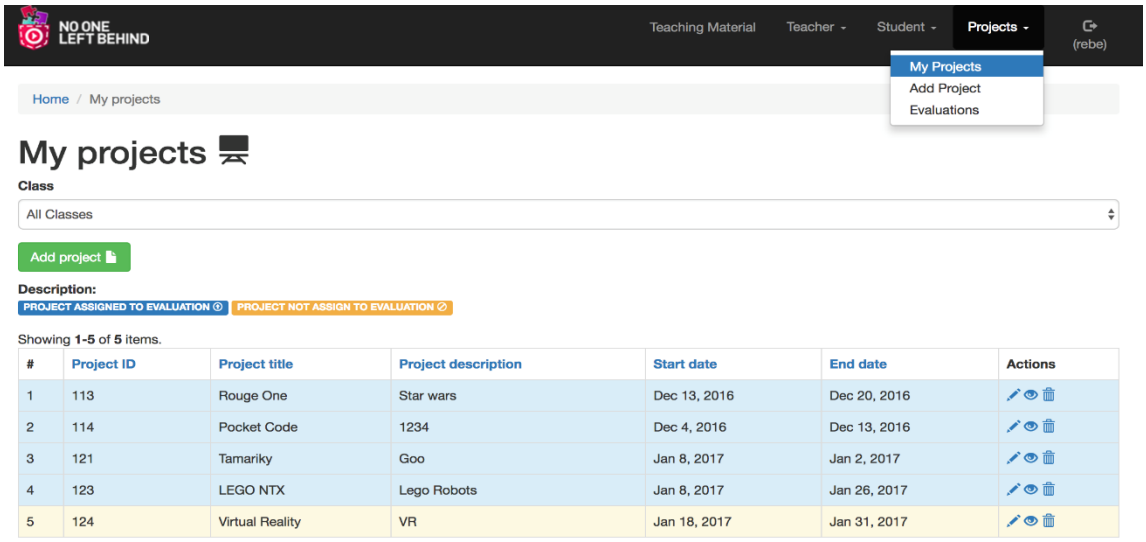


Figure 60: Projects tab – My projects view

The evaluation's tab is the most important process of the PMD because it provides teachers an efficient way to test and to evaluate the student's work.
Once the evaluation view is open, the teacher is able to review the classwork. Therefore the student must submit their program through the Pocket Code web-share thus give the teachers the option to play their game in a Web player, through a link in the evaluation view.
The process to evaluate a program is very easy and flexible for the teachers because they can evaluate each one of the students' program by assigning values to a set of nine parameters, with a scale from one to ten that he or she considers relevant for the evaluation of the student. After saving the evaluation the teacher will know the average score of the student's evaluation. These observable parameters are explained in more detail in Delivery 4.3.

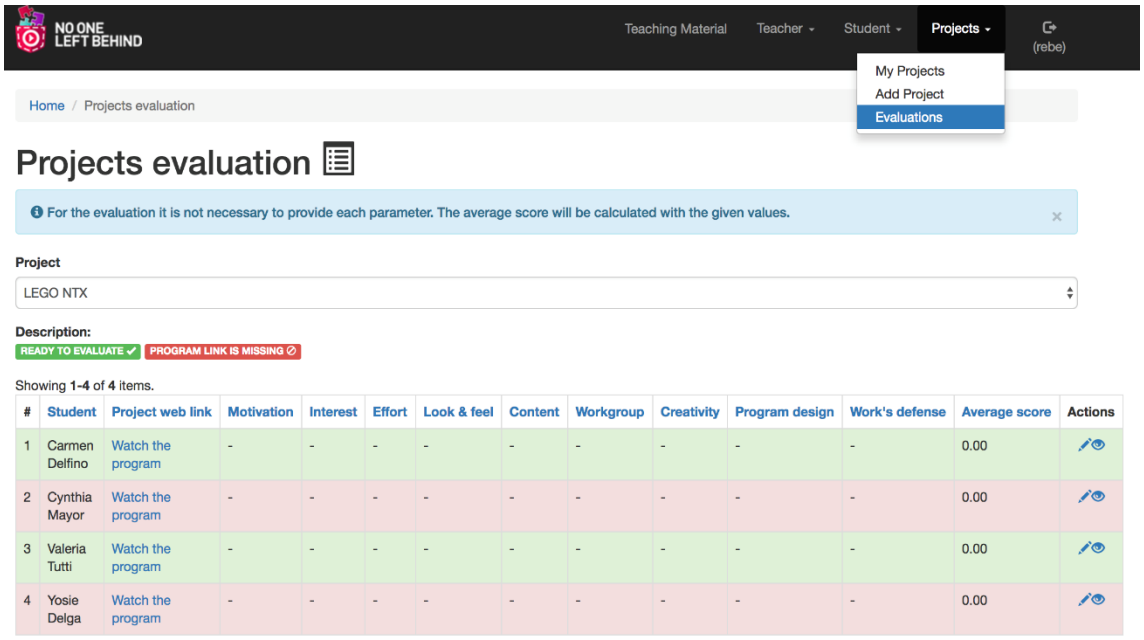


Figure 61: Evaluations tab – students' evaluation list

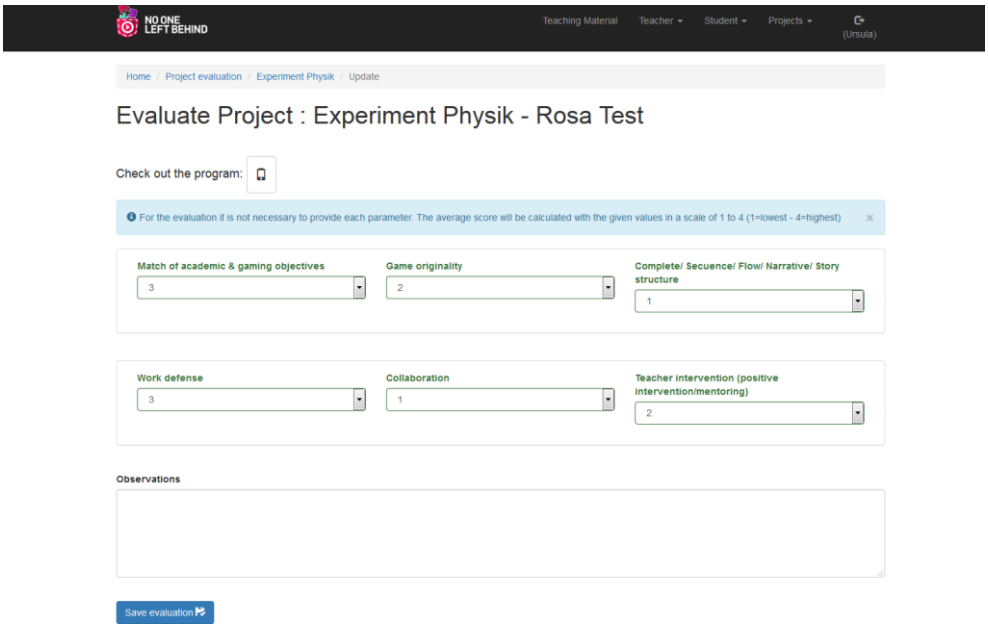
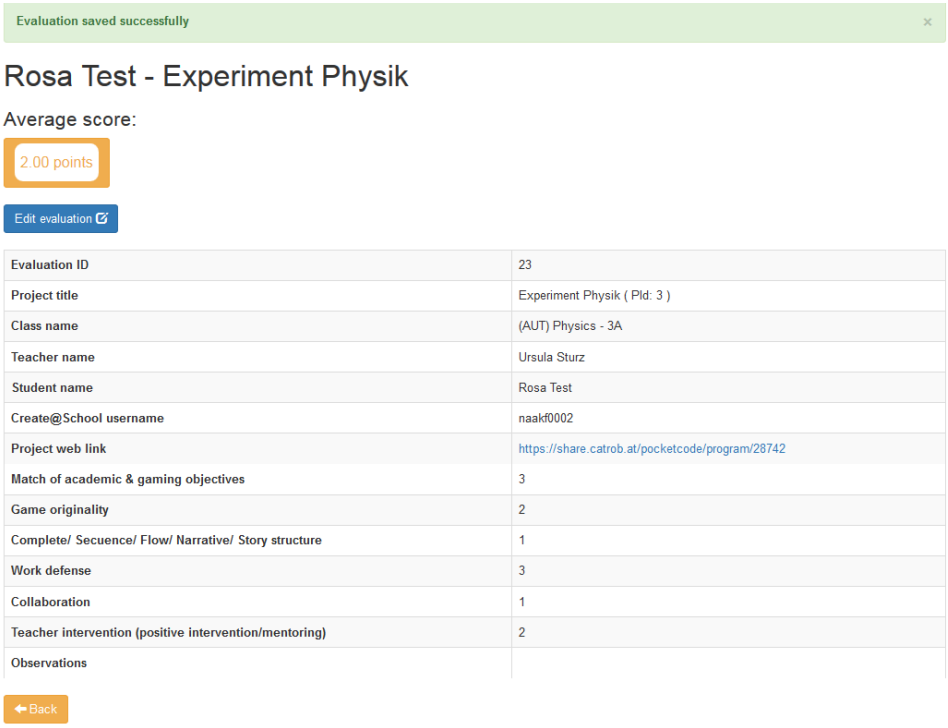


Figure 62: Evaluations tab – evaluation form



Evaluation ID	23
Project title	Experiment Physik (PId: 3)
Class name	(AUT) Physics - 3A
Teacher name	Ursula Sturz
Student name	Rosa Test
Create@School username	naakf0002
Project web link	https://share.catrob.at/pocketcode/program/28742
Match of academic & gaming objectives	3
Game originality	2
Complete/ Secuence/ Flow/ Narrative/ Story structure	1
Work defense	3
Collaboration	1
Teacher intervention (positive intervention/mentoring)	2
Observations	

Figure 63: Projects tab – evaluation results

3.1.5 Project submission through web-share

All NOLB-users have now a button within the detail page of their programs, see figure 64.

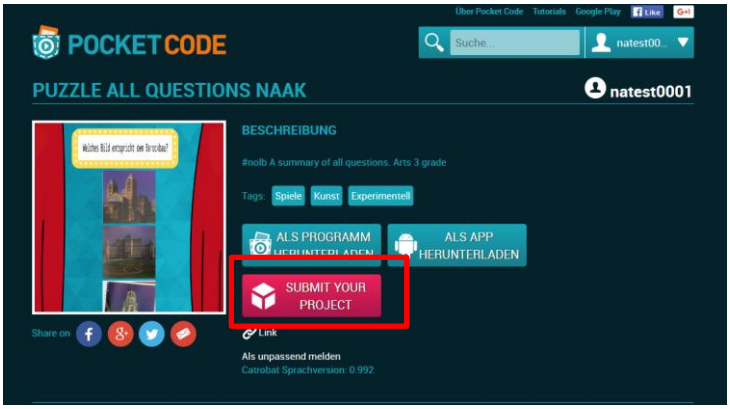


Figure 64: Button to submit a project

When tapping on this button students get asked to provide a project ID, see figure 65. They get this id from their teachers. If the project ID exists a success message is returned. Otherwise the system will return an error message.

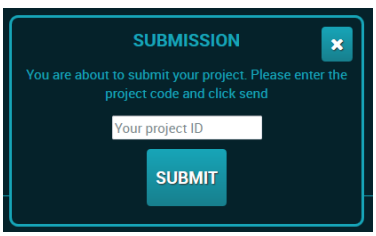


Figure 65: Submission process with project ID given by the teacher

3.2 Dedicated storage space for NOLB

On our edu platform (https://edu.catrob.at/no1leftbehind) students, teachers and parents find an own dedicated space for them. The new site targets not only the innovative teachers but also the parents and students themselves. We have planned to provide each target audience with their own materials They get information to NOLB, Create@School and find here all descriptions (e.g., how to use the apps) and tutorials (basic steps as well as tutorials), see Figure 66.

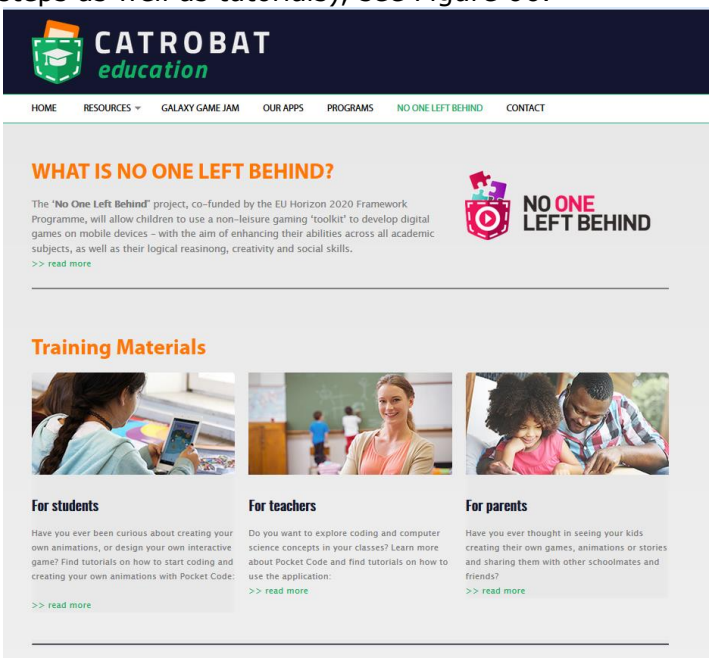


Figure 66: NOLB dedictated space

Further they find pages that support game jam events or other events like hour of code, tutorials, beginner course, brick documentations etc. see Figure 67.

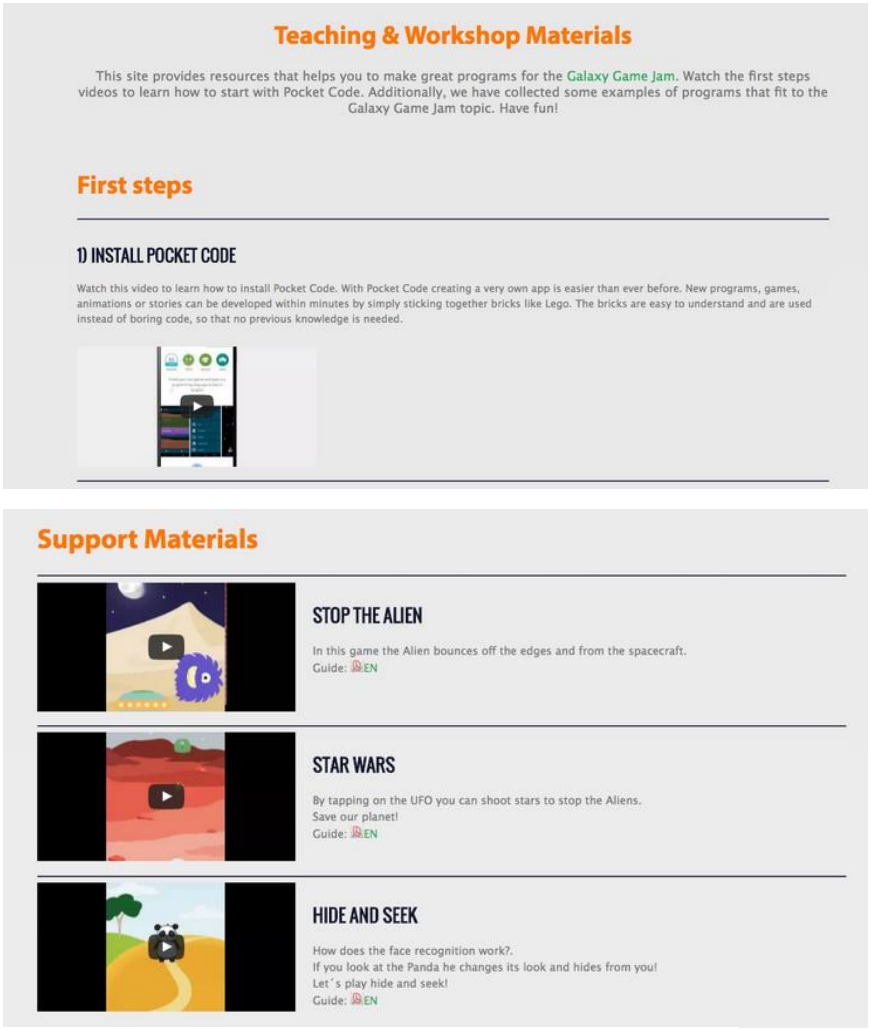


Figure 67: Own NOLB section within the edu-platform

The redesign of this space included:

- New menu structure
- Featuring information about the Create@School app
- Featuring new template instructions
- New tutorial cards
- Teacher Guides

See Figure 68.

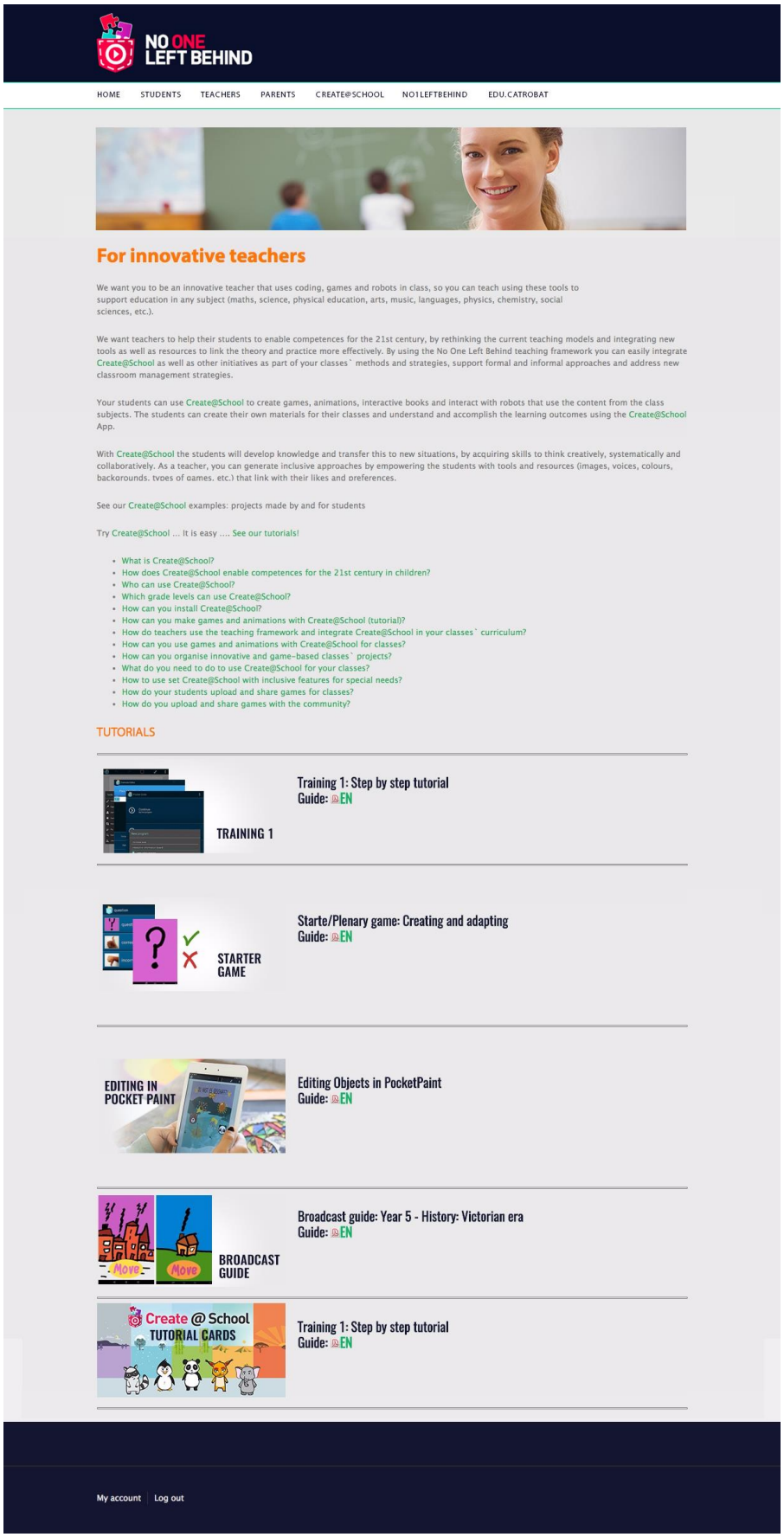


Figure 68: Subpage with materials for teachers

The new section is publicly available since January 2017.

3.2.1 Web-share features for teachers

Within our web-share new functionalities especially for teachers have been implemented.

Online Code view: This feature provides a moderator overview over program content (also scripts, looks and sound, flat one page view). This "flat" view of the programs for user enables to see all scripts and images and sounds in one large page.

Set programs to private: Users have now the possibilities to set their shared programs to private. This programs are not displayed on the main page of the web view either can be searched for. Programs can therefore only be shared with people who have the link. Therefore it's easier for teachers to create their games without students seeing them on the web view.

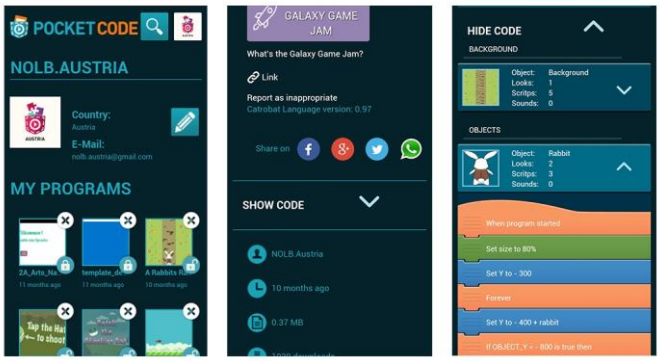


Figure 69: New functionalities web-view

Code-statistics: Below the "Show Code" option there is now a ne option with the name "Code Statistics". Teachers can now see the number of scripts, bricks, objects, etc. (without downloading the program itself), see Figure 70.

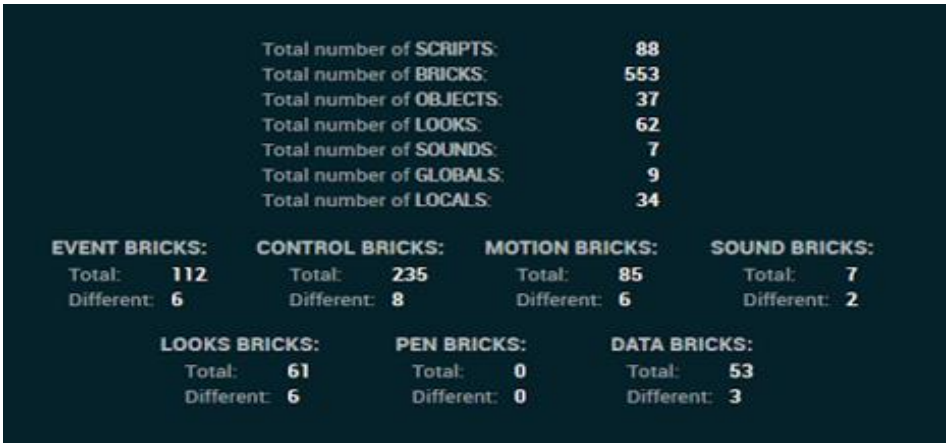


Figure 70: Code statistics for assessment of programs

3.3 The Analytics Engine

The Analytics Engine is made up of several components and the data visualization component (dashboard) is the last one in the chain. The following figure shows the overall architecture.

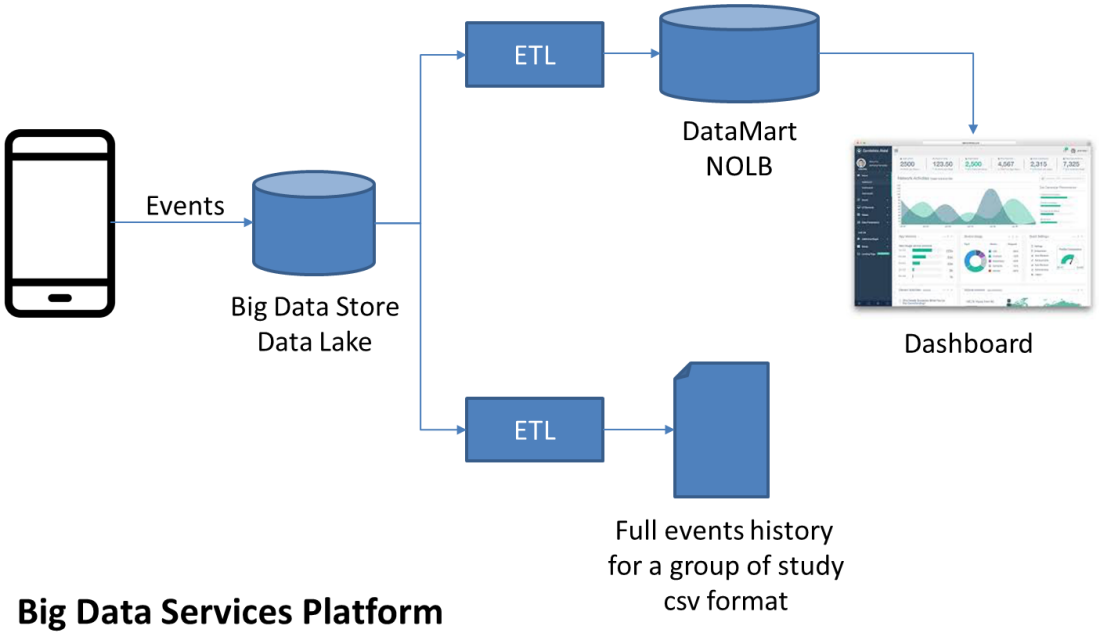


Figure 71: Big Data Services Platform

3.3.1 NOLB Data tracking

The Create@School app makes use of an SDK, the BDSClientSDK (Big Data Services Client SDK), which is a very simple and lightweight library with no external dependencies that allows developers to send different types of events related with their applications to the Big Data Services (BDS) platform. The BDSClientSDK library is a .jar file that gets included in the libs folder, when developing. The SDK includes information about the current device in each event.

Different types of events can be generated:

- Init session (init_session)
- End session (end_session)
- Purchase (purchase)
- Connection error (connection_error)
- Custom (type specified by developers)

One very important aspect is the management of session information. BDSClientSDK is able to detect when the app is started or pushed to background and determine if an *initSession* or an *endSession* event should be sent.

A BDSClientEvent object has the following fields (1):

- `appId`: application identifier. This value is set automatically.
- `type`: the class of the event. It could be a custom string or any of these constants:
  - BDSClientEvent.TYPE_INIT_SESSION
  - BDSClientEvent.TYPE_END_SESSION

    - BDSClientEvent.TYPE_PURCHASE
    - BDSClientEvent.TYPE_CONNECTION_ERROR
- `userId`: the user linked to the event. If no value is filled, it will be autogenerated.
- `timestamp`: the time associated to the event. If indicated value is 0, the final one will be autogenerated at the moment of event creation.
- `sessionDuration`: only for **endSession** events. Time in milliseconds.
- `customData`: optional `JSONObject` that contains additional information about the event.
- header: JSONObject that contains data about the device. Typical metrics used by **BDS** are: device_id, os, app_version, os_version, device, model, locale, screen_width, screen_height, ip, auto_time, …

Using the SDK, with every event the following parameters get tracked:

Table 6: Parameters of a log

Parameter	Description
appId	NOLBPocketCode
userId	username
timestamp	
event type	e.g. createExampleProgram
header	e.g. device_id, os, app_version, os_version, device, model, product, country, sim_country_iso, locale, screen_width, screen_height
custom data	e.g. name of the program, object etc.

The `customData` parameter holds the main power for tracking activity when using the Create@School app. The table 4 provides an overview of the current possibilities of the tracked data. The user input shows the performed action of the user and which custom data gets stored within the event.

Table 7: User input and tracked custom data

Nr.	User Input	Custom data
1.	User creates a new empty or example program	programname, landscape, exampleProgram
2.	User creates a new object with Pocket Paint, Camera, device, Media Library	programname, scenename, objectname, source
3.	User creates a new look with Pocket Paint, Camera, device, Media Library	programname, scenename, objectname, lookname, source
4.	User adds a new sound with record, device, media library	programname, objectname, soundname, source, length
5.	User adds a new brick	programname, scenename, objectname, brickcategory, brickname, brick position
6.	User adds variables or lists	programname, scenename, objectname, variablename, scope
7.	User deletes a program	programname
8.	User deletes an object,	programname, scenename, objectname, scriptsamount, bricksamount, looksamount, soundsamout
9.	User deletes a sound, look, brick, variable or list	rogramname, scenename, objectname, name
10.	User copies program, object, look, brick, sound	programname, scenename, objectname, name
11.	User opens program	Programname
12.	User clicks on explore	Duration of session
13.	User clicks on help button	Duration of session
14.	User downloads a program	Programname
15.	User uploads a program	Programname, description
16.	User uses templates	Templatename, landscape

17.	User changes GPII settings	Profilename, settingName
18.	User clicks on Bricks-Help option	programname, scenename, objectname, brickname
19.	User adds, deletes or merges scenes	programname, scenename
20.	User creates group	programname, groupname
21.	User uses formula	programname, scenename, objectname, brickname, brickfield, formula
22.	User executes stage	Programname, scenename, duration of session
21.	User use Pocket Paint	Duration of session
22.	User uses backpack (unpack/backpack)	programname, scenename, objectname

The Analytics engine uses the concept of Data Lake for storing information. This concept is independent of the format of the data, which was a typical limitation when using a purely relational database format. Data "registered" from the Create@School app is making use of Amazon Web Services, specifically an Amazon S3 Storage instance, that can grow elastically as needs for data stored grow.

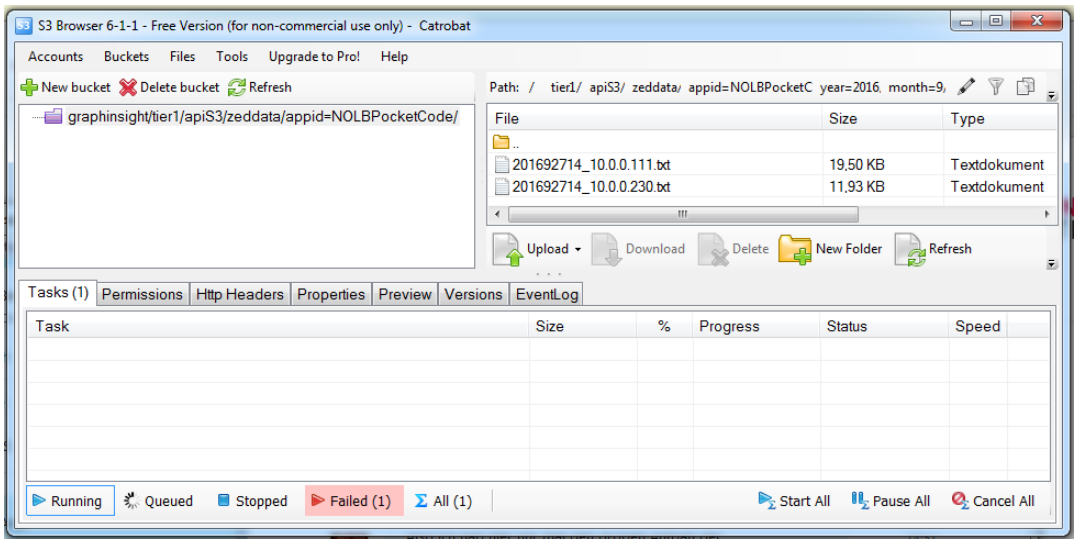


Figure 72: Amazon S3 Storage.

Example log:

[{"appId":"NOLBPocketCode","**userId":"Nolb.Austria**;1474987943656;c**reateExampleProgram**;device_id#ac73e9722742481f|os_version#|session_duration#0;amount#|msisdn#;at;Android;1.0;en_GB;;
;**custom_data#{"programName":"bird","landscape":"true","createExampleProgram":"true"}**
;

standard_data#;header#{"device_id":"ac73e9722742481f","os":"Android","app_version":"1.0","os_version":"3.10.84-perf-g5c59022","version_release":"6.0.1","device":"clark","model":"XT1572","product":"clark_reteu","brand":"motorola","connection_type":"wifi","carrier":"","network_type":"lte","country":"at","sim_country_iso":"at","mcc_mnc":"23201","display":"MPHS24.107-58-1","hardware":"qcom","id":"MPHS24.107-58-1","manufacturer":"motorola","locale":"en_GB","screen_width":1440,"screen_height":2392,"screen_dpi_density":560,"rooted":false,"allow_mock_location":"0","ip":"129.27.34.129","adb_enabled":"1","airplane_mode":"0","auto_time":"1","bluetooth_on":"0","data_roaming":"0","development_settings_enabled":"1","non_market_apps":"1","usb_mass_storage_enabled":"1","wifi_on":"1"};

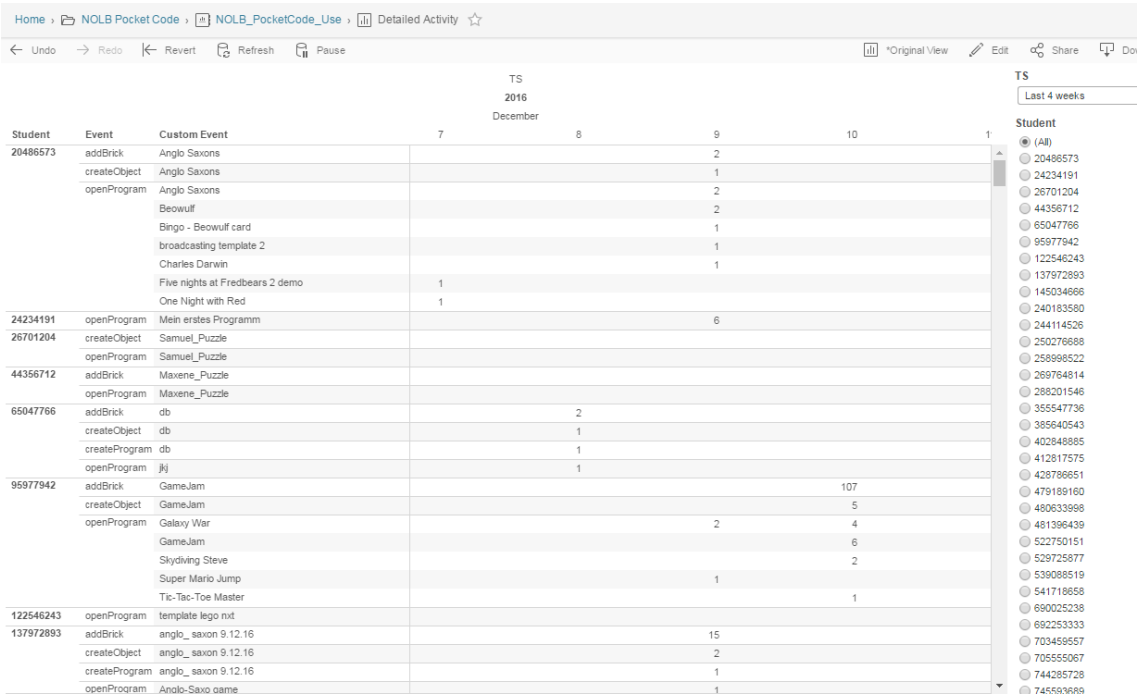
A lot of information can be stored. The following is just a sample:

Table 8: Sample data

User-Input	Parameters	Pocket Code	Data (only customData)
user creates new empty program	eventType (CreateProgram) userId (username) timestam (systemtime customData (programname, landscape, exampleProgram)	Class: ProjectManager Method: initializeNewProject	"custom_data":{"programName":"trackMe","landscape":"false","createExampleProgram":"false"}
user creates new object with Pocket Paint	eventType (CreateObject) userId (username) timestamp (systemtime customData (programname, scenename, objectname, source)	Class: NewSpriteDialog Method: handleOkButton	"custom_data":{"programName":"trackMe","sceneName":"Scene 1","objectName":"firstObject","source":"PocketPaint"}
user deletes program	eventType (DeleteProgram) userId (username) timestamp (systemtime) customData (programname)	Class: ProjectListFragment Method: deleteCheckedProjects	"custom_data":{"programName":"trackMe3"}
user deletes object	eventType (DeleteObject) userId (username) timestamp (systemtime) customData (programname, scenename, objectname, scriptsamount, bricksamount, looksamount, soundsamount)	Class: SpritesListFragment Method: deleteCheckedSprites	"custom_data":{"programName":"trackMe","sceneName":"Scene 1","objectName":"Blue duster-1","amountOfBricks":"0","amountOfScripts":"0","amountOfLooks":"1","amountOfSounds":"0"}
user copy program	eventType (CopyProgram) userId (username) timestamp (systemtime) customData (programname)	Class: CopyProjectTask Method: doInBackground	"custom_data":{"programName":"trackMe2"}
user copy object	eventType (CopyObject) userId (username) timestamp (systemtime) customData (programname, scenename, objectname)	Class: SpritesListFragment Method: copySprite	"custom_data":{"programName":"trackMe","sceneName":"Scene 1","objectName":"Blue duster-1_Copy","name":"Blue duster-1_Copy"}
user opens program	eventType (OpenProgram) userId (username) timestamp (systemtime) customData (programname)	Class: StorageHandler Method: loadProject	"custom_data":{"programName":"adventureTemplate"}
user clicks on explore button (web-share opens)	eventType (StartExploreSession) userId (username) timestamp (systemtime) customData (null)	Class: MainMenuActivity Method: handleWebButton	"custom_data":{}

For the preliminary version of this Delivery (D 4.1) a number of dashboards have been created to show some basic information. Several pilots have provided enough data to be visually presented. Here are some of the results.

Detailed Activity (all) – all activities from all the students can be explored in a very rough way (similar to the contents of an Excel sheet), see Figure 73.



Student	Event	Custom Event	7	8	9	10	1
20486573	addBrick	Anglo Saxons			2		
	createObject	Anglo Saxons			1		
	openProgram	Anglo Saxons			2		
		Beowulf			2		
		Bingo - Beowulf card			1		
		broadcasting template 2			1		
		Charles Darwin			1		
		Five nights at Fredbears 2 demo	1				
		One Night with Red	1				
24234191	openProgram	Mein erstes Programm			6		
26701204	createObject	Samuel_Puzzle					
	openProgram	Samuel_Puzzle					
44356712	addBrick	Maxene_Puzzle					
	openProgram	Maxene_Puzzle					
65047766	addBrick	db		2			
	createObject	db		1			
	createProgram	db		1			
	openProgram	jkj		1			
95977942	addBrick	GameJam				107	
	createObject	GameJam				5	
	openProgram	Galaxy War			2	4	
		GameJam				6	
		Skydiving Steve				2	
		Super Mario Jump			1		
		Tic-Tac-Toe Master				1	
122546243	openProgram	template lego nxt					
137972893	addBrick	anglo_ saxon 9.12.16			15		
	createObject	anglo_ saxon 9.12.16			2		
	createProgram	anglo_ saxon 9.12.16			1		
	openProgram	Anglo-Saxo game			1		

Figure 73: Basic information about detailed activity.

3.3.2 NOLB Dashboards / Visualisation of the data

In Delivery 4.3 a methodological framework has been developed to create and calculate behavioural measurements. Visualization dashboards have been developed and linked to Create@School, to support the teachers 'framework and allow visualization of performance (representation of gamified academic content). Two types of visualization dashboards have been provided: online (big data software - Tableau- supported) and offline (excel sheets). Both dashboard can dynamically update the different behavioural constructs or categories when selecting different variables such as teacher, his/her class or a student.

In the first version (December 2016) we were able to explore information about users and the sessions they have opened. Activities were grouped by:

- Users
- Amount of sessions opened
- Length of individual sessions

For instance we can see all the information for the last month. In orange all new users are presented ("new users" are those users with no previous registry in the Data Lake); in blue we see those users that have opened a session previously (one month before or two months before, for instance).

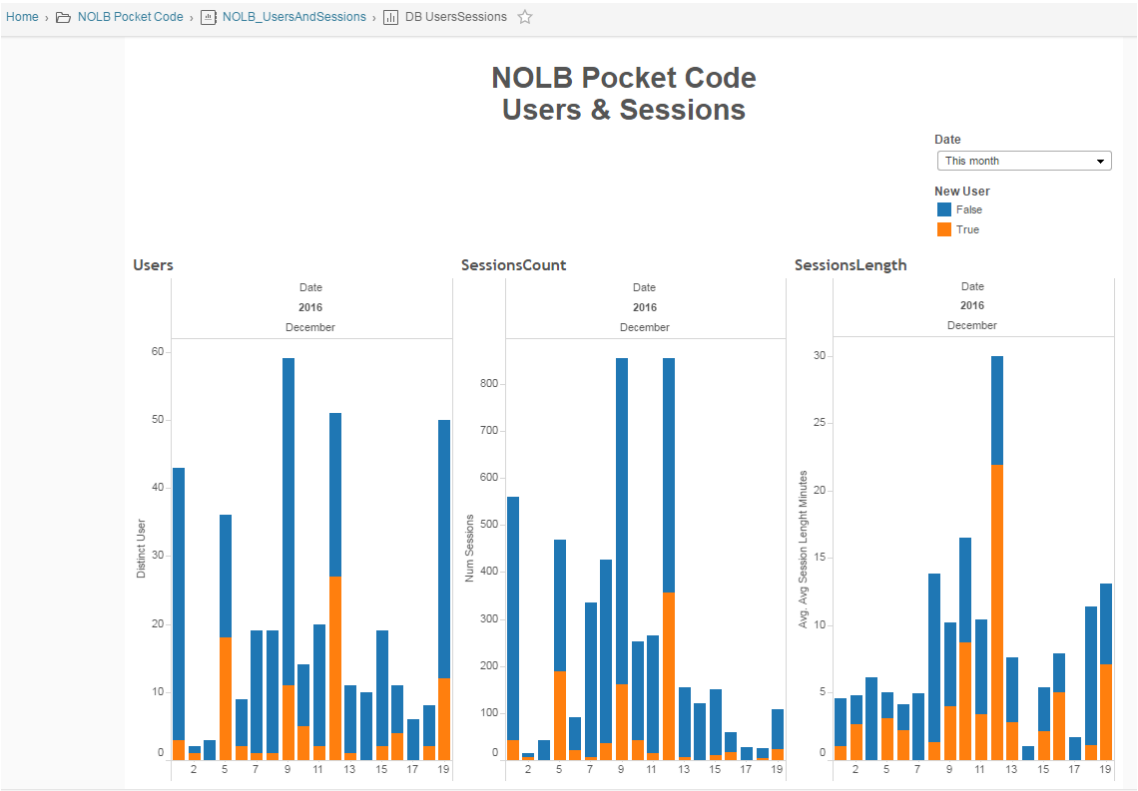


Figure 74: NOLB Pocket Code users and session in December

Or we can explore the activities in the previous 3 months. In this particular example we can see which dates group the more users connected, and the increase in activity during the month of December. This can give an initial idea about what and when to focus further analysis.

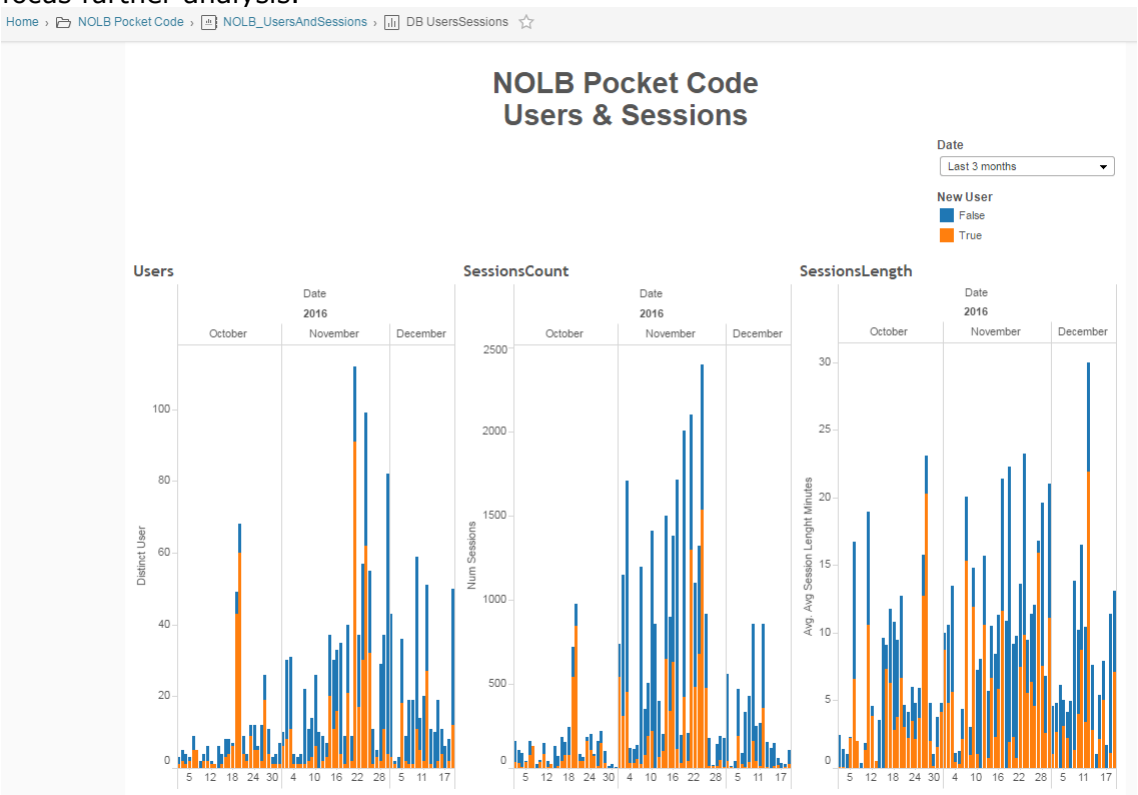


Figure 75: NOLB Pocket Code users and session in October, November and December

NOTE: the timeframe of analysis can be expanded or narrowed as desired, from focusing in a session of one hour to the activity recorded in a full year. It will depend on the user to choose the appropriate timeframe to dig deeper.

We can then check the information associated to an individual student. If we explore how many times a student uses a given action in a given session, we can reach some interesting conclusions. The following dashboard shows the amount of times the action *openProgram* is used by all students (for simplicity we have reduced the timeframe to December)

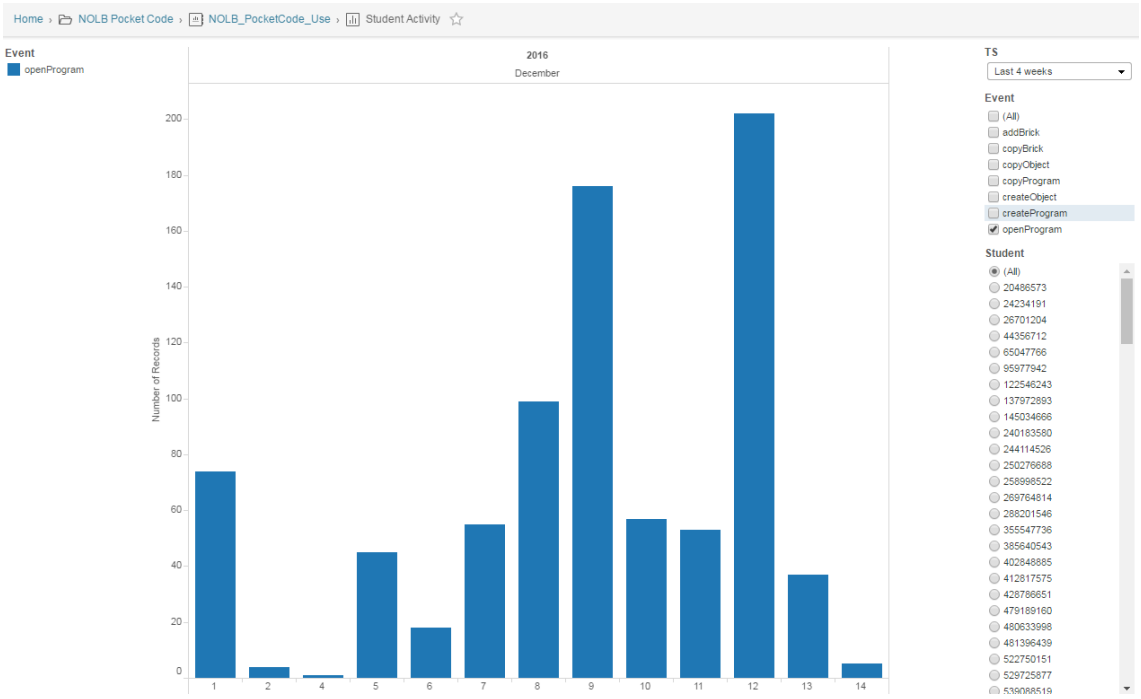


Figure 76: Amount of times the action *openProgram by all students in December*

Or we can narrow the information about that same action to a single student:

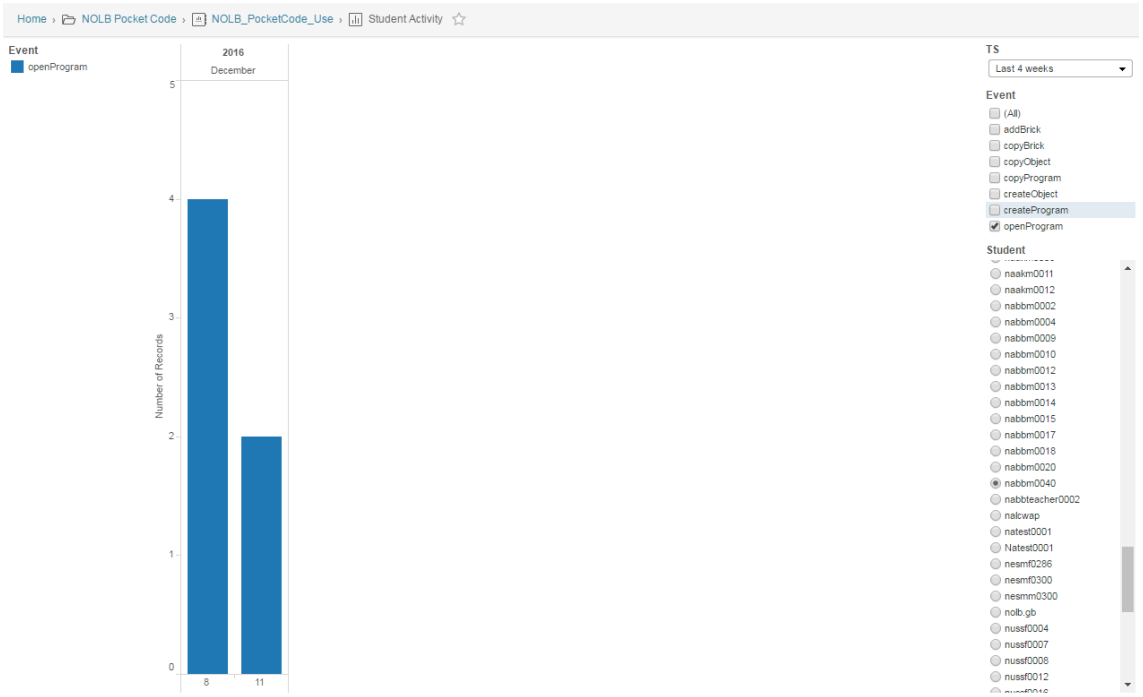


Figure 77: Amount of times the action *openProgram by one student in December*

We can explore a combination of actions at the same time. Here we select *openProgram*, *createProgram*, *createObject*, and *addBrick*:

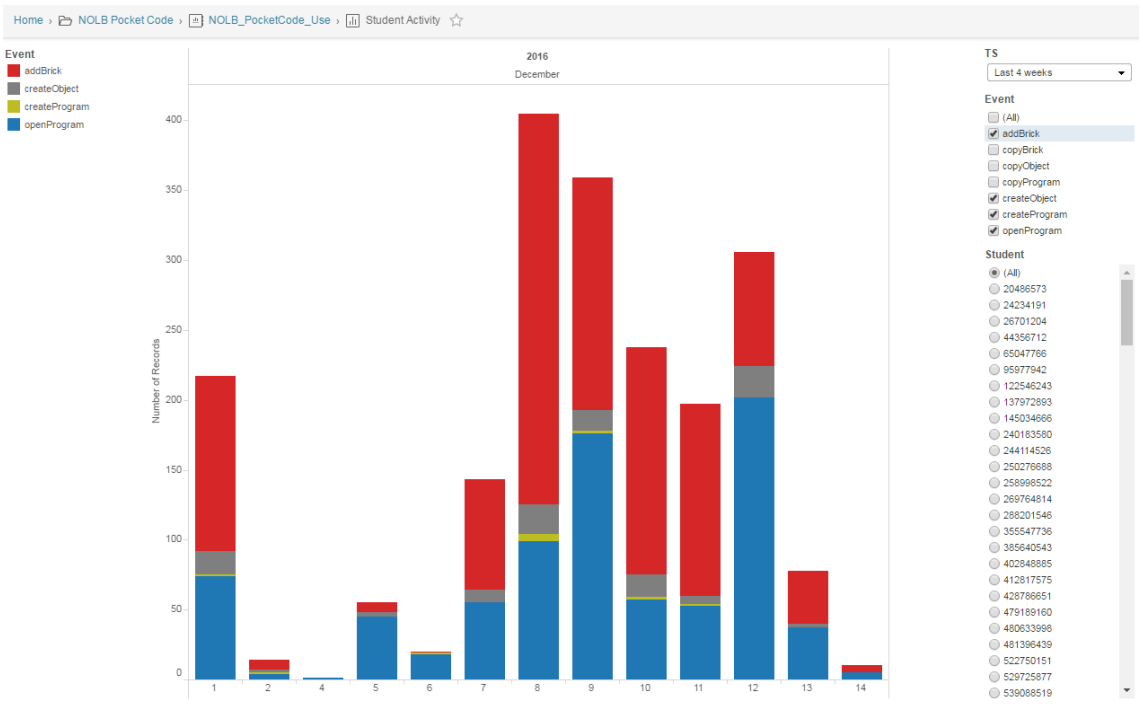


Figure 78: Tracked data visualized: addBrick, createObject, createProgram, openProgram by all students

All of these actions can be the basis for a measurement of creativity of a single student or a group of students.

Or we can explore the activity of one single student for the full collection of actions (these is a sample and they can be extended to cover the full list):

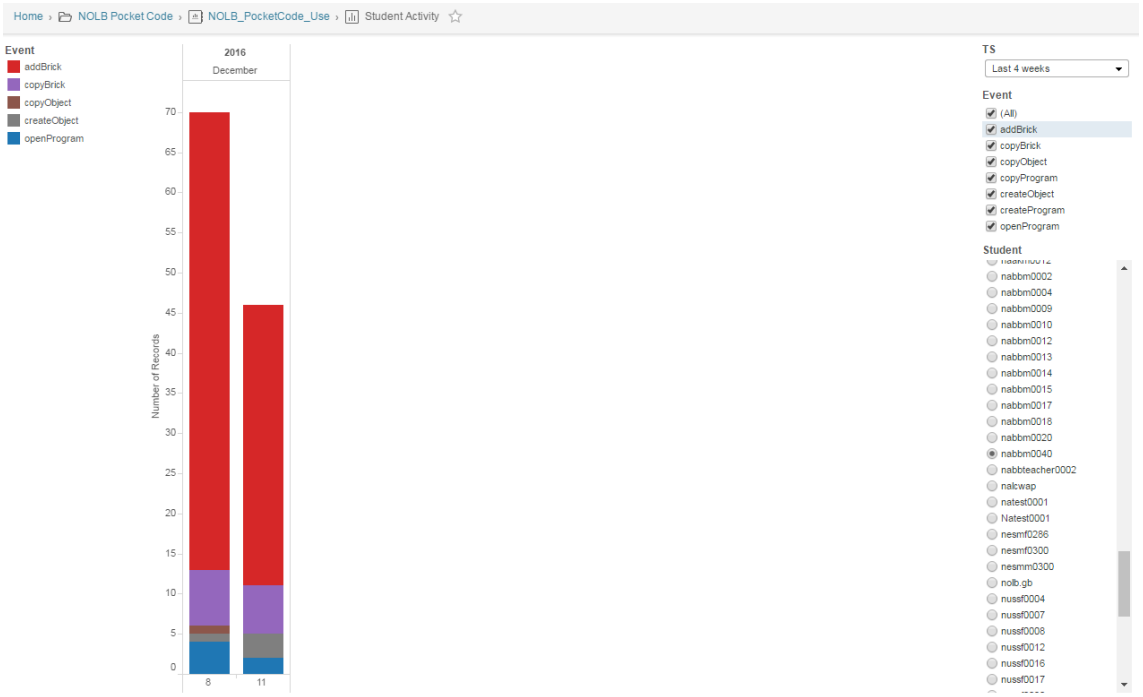


Figure 79: Tracked data visualized: addBrick, createObject, createProgram, openProgram by one student

One final feature that is very relevant is the dashboard for establishing comparisons. There are just two basic possible comparisons: many-to-one and one-to-one. In the first case the "behaviour" of the whole group is compared to one individual. In the second case the "behaviour" of one individual (just any one) is compared to another individual.
In the following dashboard we compare the group using two actions (*openProgram*, *addBrick*) to those same actions used by one particular student.

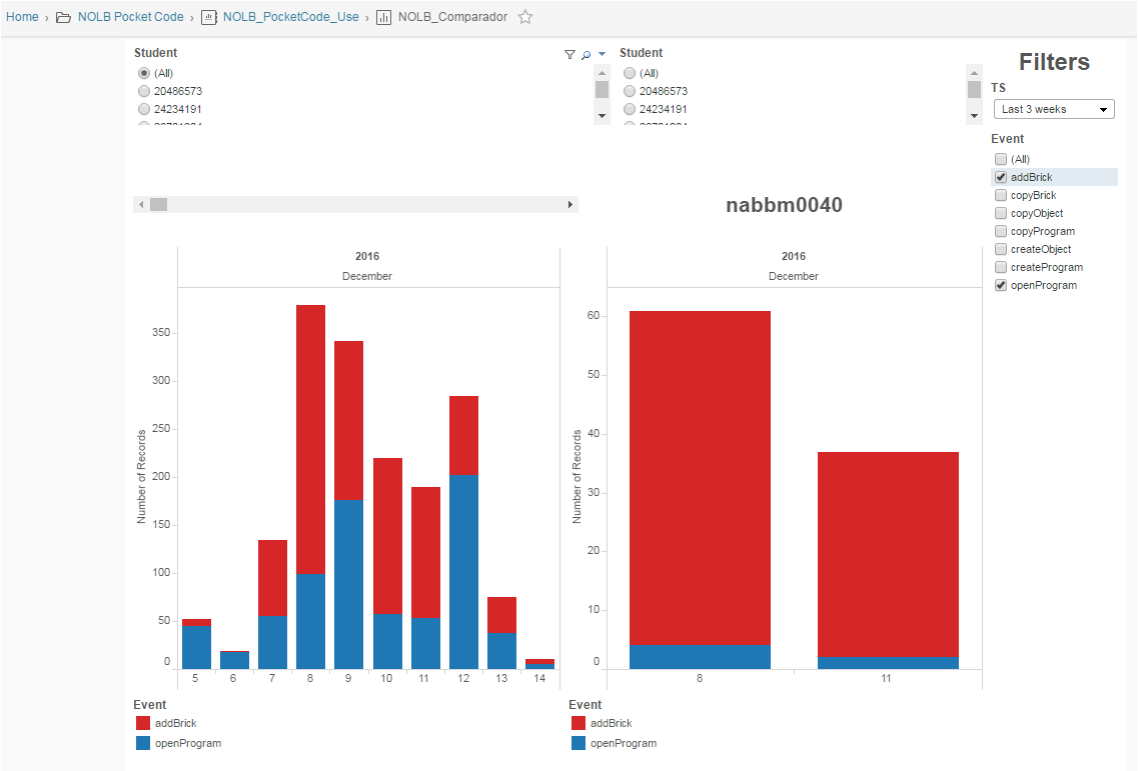


Figure 80: Similar comparison (same actions) between a group and one particular student

A similar comparison (same actions) can be explored for two individual students.

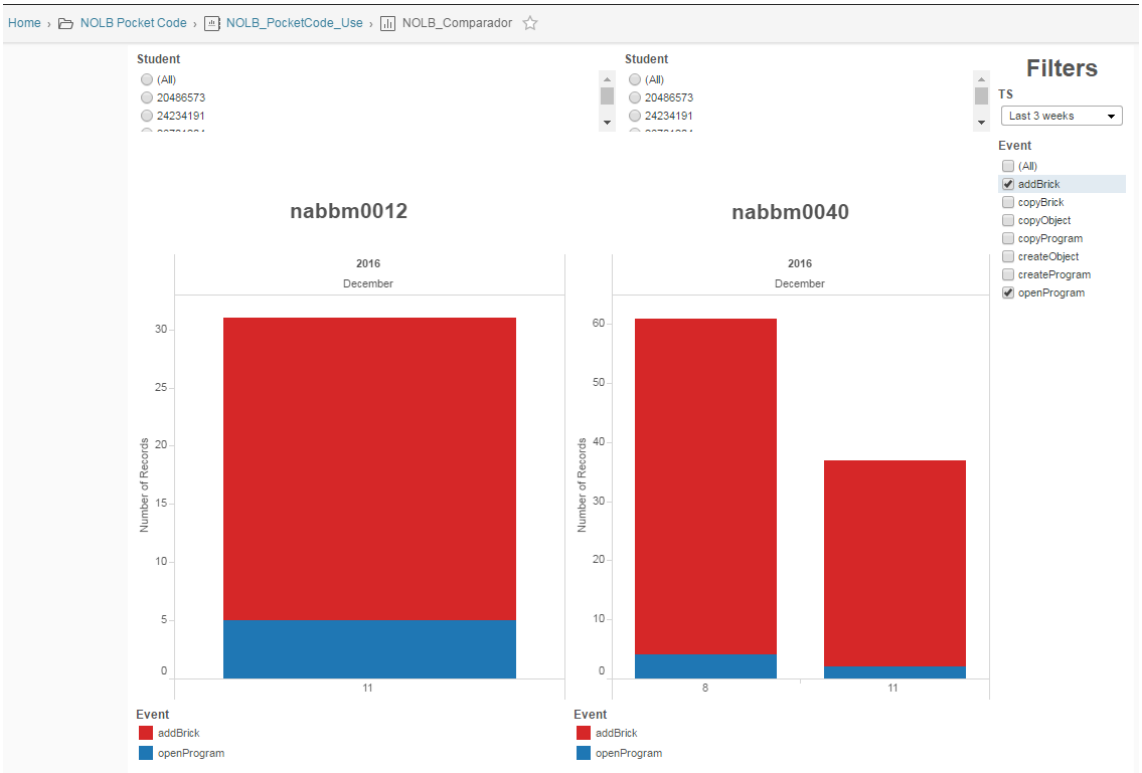


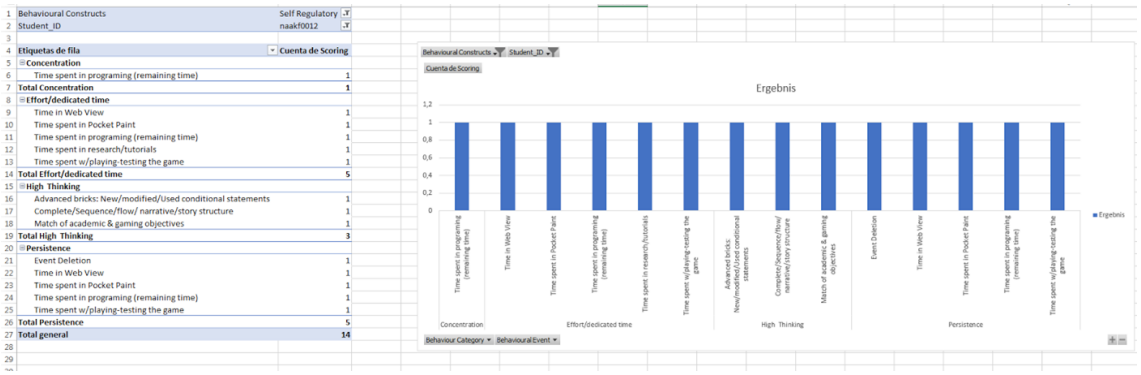
Figure 81: Similar comparison (same actions) between for two individual students

This tables and possibilities allowed us to develop the methodological and technical requirements to link the visualisation dashboard with the data captured while using Create@School and the data observed by teachers provided trough the PMD. This was part of Delivery 4.3.

The information observed by the teachers captures the perception of the teachers regarding the academic readiness and achievement of learning goals (e.g. reaching the academic goal) and some students 'abilities and social behaviours (such as creativity and collaboration). The following can be assessed by the teacher: Matching of the academic and gaming objectives, Game originality, How well the game works, Defence of the work done or submitted project, Collaboration, and Teacher intervention.

The automatic collection of data through the SDK component of the analytic architecture provides evidences, as well as continuous tracking on the activities and usage of coding components handled by the student to develop games for their lessons. The events have been categorized and form the following parameters: Coding Skills, Look & Feel / Customisation / Aesthetics, Time management parameters and Support parameters.

Startng in January 2017 we developed concrete dashboards for data visualization, after receiving feedback from interested parties, mainly the teachers that would assess the students' usage of the Create@School app. The definition of the behavioural and academic measurements have been developed together with the partners in Spain and TUGraz in iterative steps. Figure displays a first outcome.



Behavioural Constructs	Self Regulatory
Student_ID	naakf0012
Etiquetas de fila	**Cuenta de Scoring**
Concentration	
Time spent in programing (remaining time)	1
Total Concentration	**1**
Effort/dedicated time	
Time in Web View	1
Time spent in Pocket Paint	1
Time spent in programing (remaining time)	1
Time spent in research/tutorials	1
Time spent w/playing-testing the game	1
Total Effort/dedicated time	**5**
High Thinking	
Advanced bricks: New/modified/Used conditional statements	1
Complete/Sequence/flow/ narrative/story structure	1
Match of academic & gaming objectives	1
Total High Thinking	**3**
Persistence	
Event Deletion	1
Time in Web View	1
Time spent in Pocket Paint	1
Time spent in programing (remaining time)	1
Time spent w/playing-testing the game	1
Total Persistence	**5**
Total general	**14**

Figure 82: Analytics engine: visualisation in excel

Beside the description of the methodology in Delivery 4.3 the visuals have been evaluated per country as well. The differences e.g., between subgroups (gender) and classes are part of Delivery 5.3 to 5.4 Report and findings from experimental pilot UK/Austria/Spain.

4 CONCLUSIONS

This deliverable summarize all features, implementations, extensions and new functionalities of the educational app Create@School, see Figure 74. Create@School represent now a more game-based oriented, user-friendly and accessible version by connecting gaming mechanics and dynamics with pedagogical and academic methodologies, as well as the inclusion requirements through the GPII infrastructure.

By tracking certain events and user inputs we are able to provide measured and analysed data (both academic and behavioural oriented data) as statistics/visuals to teachers to help them to evaluate students programs. The dashboards help teachers to improve the quality and efficacy of the academic curriculum.

Figure 74: Create@School environment

The goal of the NOLB project was to improve Pocket Code for educational needs and to find new possibilities to make the app more attractive for students and teachers. It is important to show students new ways of learning by providing them ways to explore, allowing creative and collaborative styles, and try to unlock their full potential. Constructionist gaming and tools that encourage learning by doing have common features: They enable working in teams, allowing to express own ideas, and provide a visual programming language that is easy to understand and to learn. With its focus on schools, Create@School has not only the potential to teach students how to code, but also to teach students technical skills and empower them all over the world.

ANNEX 1

Nr.	Task	Objectives	Progress
1.	object collision (touches) with certain other object	Needed to create simple games - IMO the most important thing in game creation!	100%
2.	object collision (touches) with certain color	This makes simple labyrinth games posisble - Vital functionality in mini-game creation. E.g. skateboard games or games like "Marian!" can only realized meaningfully with this feature.	30%
3.	"Speak" and "think" word balloons, with and without timeout (a simple way to manipulate input and output variables)	"Makes many types of interactions *much* easier, Not only that - but story telling is made easy - no extra graphics needed for text"	100%
4.	Show variable (continuously, like in Scratch)	Makes many types of interactions *much* easier	100%
5.	Landscape support	Many games use landscape and it is not possible to cast it to a projector => important for schools	100%
6.	Stamp brick	"Very important for all NOLB curricula that would benefit from being able to draw, such as math, chemistry, physics, sports, economics, geogrpahy, arts, ...	100%
7.	Clear drawings	"Important for all NOLB curricula that would benefit from being able to draw, such as math, chemistry, physics, sports, economics, geogrpahy, arts, ...WIP => have working prototype mady by XDreamTeam"	100%
8.	Single finger x,y touch position, via sensor variable	"Makes new types of games feasible, e.g., makes dragging of objects possible which makes game interaction better	100%
9.	Screen touched?, via sensor variable	"Makes new types of games feasible, Needed for finger finger position anyway otherwise it makes not too much sense, WIP => have working prototyp made by XDreamTeam"	100%
10.	Undo/redo functionality for everything; get rid of confirmation messages (e.g., "Do you really want to delete this object?" when undo is anyway available)	Important because of usability. On hold!	80%
11.	Ask brick: Text input in Stage, reply in answer program variable	Makes new types of interactions feasible	100%
12.	Shared variables, shared broadcasts	Allows multi player games	0%
13.	Sound editing in Pocket Code	Nice to have, like in Scratch 2.0 (trimming, fading etc).	70% Pocket Music
14.	If then brick (without else)	"Nice for readability and UX reasons., Makes programming easier with less useless bricks in the script"	100%

15.	Switch to previous look	Similar to "Next look", but the other way around.	100%
16.	Repeat until brick (or repeat while?)	"Nice for functionality reasons., Easy to accomplish and nice to have"	100%
17.	Wait until brick	Nice for functionality reasons	100%
18.	“Play sound until done” brick	Would be nice. This is just a similar brick like the "Start to play sound XXX" brick, but the difference to that one is that the “Play sound until done” brick *waits* until the sound has been played to its end	100%
19.	Video look (on device)	Eugenio wanted this; probably easy since we already will soon have video for Scratch 2.0 video bricks and drone anyway => "low hanging fruits".	100%
20.	Video look (streaming)	Eugenio wanted this; probably easy since we already will soon have video for Scratch 2.0 video bricks and drone anyway => "low hanging fruits".	0%
21.	Select color of pen	Important for all NOLB curricula that would benefit from being able to draw, such as math, chemistry, physics, sports,economics, geogrpahy, arts…	100%
22.	Width of pen	Important for all NOLB curricula that would benefit from being able to draw, such as math, chemistry, physics, sports, economics, geogrpahy, arts, ...	100%
23.	User defined bricks more completely, e.g., allow script to be executed as one step, without disturbance by other scripts (not perfectly implemented in Scratch, but in BYOB)	Nice to have	99%
24.	Left right, up down orientation, also no rotation, via brick	Makes simple programming much easier	100%
25.	Shade of pen, of stamp of objects	Nice to have	100%
26.	Gesture recording and recognition	Would be great	0%
27.	Subcategories in Formula editor	Makes the formuas easier to find	100%
28.	Object collections (group several objects in a folder, e.g., "opponents" or "second level") - two (or three?) level only, or arbitrary hierarchy?	makes programming easier; Jonathan wants this. Very useful for Game Jams.	100%
29.	Graphical hints (simple; complex, e.g., animated)	Improvement of UX for all aspects of Pocket Code adoption, including in the NOLB context.	0%
30.	Reordering of objects	Very important because of usability.	100%
31.	Renaming variables	Important because of usability.	100%
32.	Clone bricks and paste everywhere in Pocket Code	Nice to learn about object oriented, programming, and it makes game, development much easier, Backpack?	100%

33.	Position bricks: set x and y by moving look on stage; same for rotation and size	Highly desirable because of usability.	0%
34.	Reordering of looks	Very important because of usability.	100%
35.	Delete event bricks but leave other bricks (aka commenting out parts of code)	Important because of usability.	100%
36.	Swipe gesture to switch between program overview and current program, between scripts and stage	Very important because of usability.	50%
37.	Debug mode (watch scripts running, inspect variables and lists while program is running)	Highly desirable because of usability	50%
38.	Project details (description, author, link to sharing site, screen resolution)	Would be great	100%
39.	Rename program while it is the current program	Would be nice	100%
40.	Modify the way of deleting (dustbin icon)	Important because of usability, remove it from the menu bar	100%
41.	Some way to see truncated text (currently text is cut and ... is shown at end), e.g., when names of objects or broadcasts etc are too long for one line	Important because of usability	70%
42.	When coming back to scripts area after having switched to some other place (looks, stage, list of obejcts, ...), show same part of scripts on screen as when user left the script view. Same for list of objects, looks, sounds.	Important because of usability	100%
43.	Allow to edit strings in formula editor	Important because of usability	100%
44.	Detect the end of sounds		100%
45.	Not to allow concurrent sounds; this means that if a new brick starts a sound all the other sounds can be stopped.		100%
46.	Separate events from other control bricks	Important for kids!	100%
47.	Center point for looks (for turn bricks)	Nice to have	0%
48.	Show hints when first opening the app	Simply awesome	100%
49.	Speech recognition (event, sensor)	Nice with robots and users with visual impairments, or users with autism.	100%
50.	Virtual gamepad emulation.	Makes it easier to program games that need game pad control (this is not only interesting for Google TV game creation allowing kids to create games for their family's Google TV: was announced during Google I/O 2015, available for instance on all new Sony Bravia TVs), as it can be used for any game, even if one does not have an Android TV.	100%
51.	Multi-touch finger positions (via list)	Important for games where several fingers are needed, e.g., for zooming, or multiplayer games that need more than tapping.	100%
52.	Suport younger kids (8-12 years) with GPII	Nice for younger kids, their parents, and primary schools.	100%

53.	3D support	Nice in the future. Proof of concept for Pocket Code was done by one Master's student, works nicely. Possible integration with new version of second life currently developed by Linden Labs, which might become very influential.	0%
54.	Animated gif builder (plus truncation)	cool	0%
55.	Camera *brick*: allows to take a picture with front or back camera during program execution, and store as temporary next look (later: save to program, to file system, save to cloud?)	Would be nice	0%
56.	Create Live Wallpaper	Interesting for social reasons. Might be of some importance for girls version. Work in progress.	70%
57.	New brick to detect background switches		100%
58.	Allow merging of programs	Cool for grop work and game jams	100%
59.	Drag'n'Drop for objects, looks, sounds and NFC tags	usability	100%
60.	Object recognition (incl. machine learning) (simple: 2d, later also 3d)	Good for projects with robots.	0%
61.	Face recording (video) for storytelling	Would be great	0%
62.	Scratch to Catrobat converter	Search for Scratch games online and convert them directly within the app	100%
63.	"When 1<2 is true" brick		100%
64.	New proberties	Object: look number and look name Device: date and time (year, month, weekday, hour, minute, second)	100%
65.	Geo location sensors	Latitude, longitude, location_accuracy, attitude	100%
66.	Ask brick and store written answer	e.g. great for quizzes	100%
67.	Integration of scenes methaphor	Create for scnes (levels) that act like separated games	100%

REFERENCES

1. Yasmin Kafai. 1995. Minds in Play: Computer Game Design as a Context for Children's Learning. Lawrence Erlbaum (1995).
2. Yasmin Kafai. 2006. Playing and making games for learning: Instructionist and constructionist perspectives for game studies. Games & Culture (2006).
3. Yasmin Kafai and Veena Vasudevan. 2015. Hi-Lo tech games: crafting, coding and collaboration of augmented board games by high school youth. In Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15). ACM (2015), 130–139. DOI: http://dx.doi.org/DOI=http://dx.doi.org/10.1145/2771839.2771853
4. Daniel SuKuenSeong. 2006. Usability Guidelines for Designing Mobile Learning portals. Mobility 06 (2006), 25–27.
5. User Experience Professionals' Association. 2005-2012. The Usability Body of Knowledge. http://www.usabilitybok.org/glossary/19. (2005-2012). Last access 13.04.2016
6. Adele Botha, Marlien Herselman, and Darelle van Greunen. 2010. Mobile user experience in a mlearning environment. In Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT '10). ACM (2010), 29–38. DOI: http://dx.doi.org/DOI=http://dx.doi.org/10.1145/1899503.1899507
7. Jaime Sanchez and Ruby Olivares. 2011. Problem solving and collaboration using mobile serious games. Computers & Education 57 (2011), 1943–1952.
8. https://scratch.mit.edu/statistics/
9. Bohyun K. (2015) Understanding Gamification, Library Technology Reports, Vol 51, No 2, pp. 17–20.
10. Hunicke, R., LeBlanc, M. and Zubek R. (2004) MDA: A formal approach to game design and game research. Pro-ceedings of the AAAI Workshop on Challenges in Game AI. Vol 4. No. 1.
11. Lee, J.H. et al (2014). Facet Analysis of Video Game Genres. In iConference 2014 Proceedings, pp. 125–139.