



- (51) International Patent Classification:
G07F 17/32 (2006.01)
- (21) International Application Number:
PCT/EP2014/054905
- (22) International Filing Date:
12 March 2014 (12.03.2014)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
1304442.5 12 March 2013 (12.03.2013) GB
14/029,318 17 September 2013 (17.09.2013) US
PCT/EP2013/069289
17 September 2013 (17.09.2013) EP
- (71) Applicant: **KING.COM LIMITED** [MT/MT]; Aragon House Business Centre, Dragonara Road, STJ3140 St. Julians (MT).
- (72) Inventors: **KNUTSSON, Sebastian**; c/o Midasplayer AB, Sankt Eriksgatan 113, S-113 43 Stockholm (SE).
STYMNE, Patrik; c/o Sankt Eriksgatan 113, S-113 43 Stockholm (SE).
- (74) Agent: **KELDA CAMILLA KAREN STYLE**; Page White & Farrer, Bedford House, John Street, London Greater London WC1N 2BF (GB).
- (81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: MODULE FOR A SWITCHER GAME

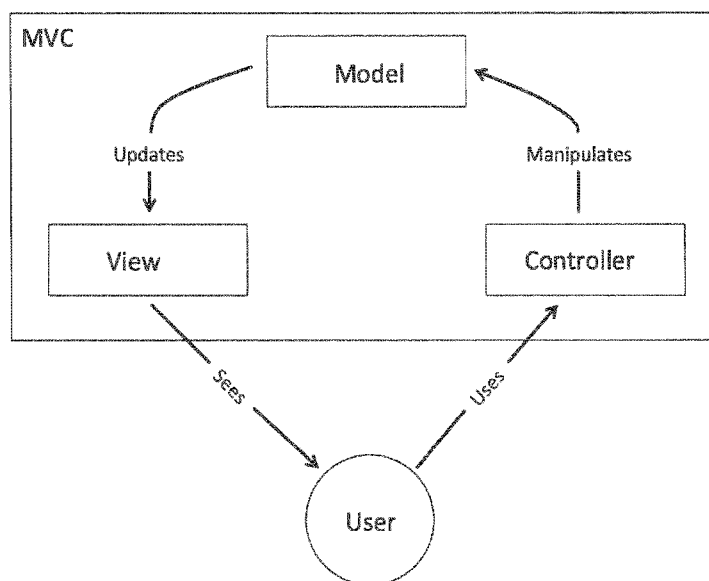


Figure 3

(57) Abstract: A method for computer game design relating to the re-use of a library of pre-existing components to reduce the development time for developing a new game in an existing genre. The MCV (model, controller, view) pattern may be used to separate functionality and allow large sections of code to be re-used without editing.

WO 2014/140144 A1



Published:

— *with international search report (Art. 21(3))*

MODULE FOR A SWITCHER GAME

BACKGROUND OF THE INVENTION

Computer implemented games is a well-known category of games that allow a player to interact with a computing device to cause the processor to perform certain calculations and typically display a result on a screen or other display device.

Different types of games have evolved from classical arcade games in to games that can be played on a handheld device such as a smartphone or personal computer. Some games are also connected to the Internet and the player can play against or compare score with other users in multiplayer mode.

10 A common genre of casual games is so-called 'match-3' games. This is a type of tile-matching game where the player manipulates tiles or game objects in order to make them disappear according to a matching criterion. In many tile-matching games, that criterion is to place a given number of tiles of the same type so that they adjoin each other. That number is often three, and the corresponding subset of tile-matching games is referred to as 'match-three games'. The criterion for matching is often to place the tiles so that a number of tiles of the same colour or of the same shape or any other characteristic are aligned or adjoined.

Games have previously been developed and programmed on an individual basis with all or most elements being built up specifically for that game. Our approach uses the model-view-controller (MVC) software architecture pattern to facilitate the creation of games of a specific type. MVC separates the representation of information from the user's interaction with it. This can be implemented in several ways, some of which are already known.

This approach drastically simplifies the way games can be built and also enhances the quality of the games created using this approach since fewer parts of the game have to be recreated for each new game of the same or similar type.

Discussion of Related Art

Casual social games have been implemented before and are known. However previous inventions have not successfully devised effective solutions to create games using a modular approach.

5 SUMMARY OF THE INVENTION

According to a first aspect of the present invention there is provided a method of designing multiple computer games, using a software module running on a processor, in which the module enables pre-defined kinds of game design functions to be implemented across multiple different computer games; and in which the
10 module implements multiple pre-defined kinds of common game design functions; and is extensible in that new components can be added to the module to create new functionality.

The module may use a Model View Controller (MVC) software architecture pattern.

The MVC may use entities, and enables components to be combined to create new
15 entities.

The common game design functions may include generating a game board of defined size and layout.

The common game design functions may include game rules or game-play logic, such as switching objects, matching objects, removing objects, and generating
20 objects.

The common game design functions may include all of switching objects, matching objects, removing objects, and generating objects.

The common game design functions may include common game rules.

The module may allow new game rules to be designed and included within the
25 module.

The common game design functions may include creating new game levels.

The module may enable debug tracking data to be stored, such as storing the board state, history of all moves.

The module may receive events that trigger the game logic to be processed and then fires off new events that can be used to draw graphics.

- 5 The method may include the step of: using data analytics to understand the impact of changes to the game design in terms of player engagement and/or monetisation and implementing changes to the game design, including frequent changes such as daily or weekly changes, to optimise player engagement and/or monetisation.

- 10 The method may include the step of deploying multiple different computer games on a game playing web site hosting a large number of computer games, the games each including the software module; using data analytics to measure player engagement and/or monetisation of a game; for games that provide sufficiently high levels of player engagement and/or monetisation, migrating those games to a platform that supports mobile game play on smartphones, tablets and also PCs.

- 15 The method may, for games that provide sufficiently high levels of player engagement and/or monetisation, the further step of using the software module to add extra game play levels and additional game design functions.

- 20 The method may include the step of using the software module to regularly add extra game play levels and additional game design functions, potentially indefinitely.

The method may be used to enable social casual games to be rapidly created and revised.

The method may be used to enable switcher games to be rapidly created and revised.

- 25 The method may be used to enable match-3 games to be rapidly created and revised.

The method may be used to enable clicker games to be rapidly created and revised.

According to a second aspect of the present invention there is provided a software module for use in at least one computer game, configured in use to run on a processor, wherein the at least one module comprises at least one component configured to provide a game function wherein said at least one component is
5 configured to be controlled by at least one parameter the game function provided by said at least one component being determined by the at least one parameter.

The module may be configured to cause at least one component to be used in one or more levels of the at least one computer game.

10

The game function may comprise at least one game element.

Said at least one parameter may be configured to control at least one of functionality, behaviour and characteristics of said at least one game element.

15

The game function may comprise a game board.

Said at least one parameter may be configured to control at least one of size and layout of the game board.

20

The game function may comprise at least one game rule.

The at least one game rules may comprise at least one of: switching objects; matching objects; removing objects; and generating objects.

25

Said at least one parameter may be configured to control at least one of the effect of the game rules and conditions for game rules to be applied.

The module may be configured to store debug tracking data for components of said
30 module.

There may be provided a method of designing at least one computer game using the module.

The method may use at least one component of said module in at least two different game levels.

According to a third aspect of the present invention there is provided a computing device comprising a model module, said module configured in use to run on a processor, wherein the at least one module comprises at least one component configured to provide a game function wherein said at least one component is configured to be controlled by at least one parameter the game function provided by said at least one component being determined by the at least one parameter.

The computing device may further comprise a view module

The computing device may further comprise a controller module.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a schematic illustration of a computing device.

Figure 2 shows an exemplary environment.

Figure 3 shows an example of a typical MVC software architecture.

Figure 4 shows an example of a 'match-3' switcher game.

Figure 5 shows an example of a 'match-3' switcher game.

Figure 6 shows an implementation of a virtual map.

Figure 7 shows a schematic layout of a user device

Figure 8 shows an example model module

DETAILED DESCRIPTION

The present invention relates to a software architecture module that can be used when creating a game. The following description will use a 'switcher' game as an example of a typical implementation of the module. It is however understood that

the invention can be implemented for other types of game such as 'clicker' and 'bubble shooter' games.

In the following description of various implementations of the invention, reference is made to the accompanying drawings which form a part thereof, and in which is shown by way of illustration various implementations in which the invention may be utilized. It is to be understood that other implementations may be utilized, and structural and functional modifications may be made without departing from the scope of the present invention.

The terms user and player are used interchangeably throughout this document and no specific meaning is intended using one or the other unless the context suggests otherwise.

Figure 1 shows a schematic picture of a computing device, containing a Central Processing Unit and Random Access Memory. The CPU acts according to input given from input devices, such as a keyboard, mouse or touchscreen. Computer BUSES are used to communicate, both between input devices and the CPU, but also between different controllers within the computer device, such as the graphics controller and the network controller. These controllers in turn communicate with external devices, such as a monitor for video output with which the graphics controller communicates, and the network controller communicates with for instance the internet, through wireless or wired connections. A user can interact with the computing device through input devices, such as a pointing device (e.g. a mouse) and a keyboard. The computing device can be static, for example a desktop computer, or mobile for example a laptop computer, a tablet or a mobile phone. The computing device can run an operating system typically associated with desktop computers, for example Windows, OSX or Linux-based operating systems, or a mobile operating, for example iOS, android or windows phone. The CPU may be operated to load an application into the RAM and run the application. The application may be a game. Alternatively, the application may be a web browser which can be operated to load and run a game program.

Figure 2 portrays an exemplary overall environment in which the present invention may be utilized. A virtual game is stored on for instance a game server 210. The

virtual game is to be played on a client device, such as a computer 240, 250 or a smartphone or other handheld device 260. The client device can also be a kiosk, arcade gaming station, smart TV or other device with computing capabilities, input devices and a screen that can present the game to a user. The client device
5 communicates with a game server 210 and a social network server 230, for instance through the Internet 220 or other network. It should be understood that the social network 230 and the game server 210 does not have to be located in different places, they could be on the same server or on a plurality of servers located in different locations. People skilled in the art will understand that other devices than
10 the exemplary ones listed can be also be used without departing from the spirit and scope of the invention.

The techniques described in this patent may be deployed in many different gameplay architectures. For example, a computer game may be implemented as a computer program that is stored and runs entirely locally on the processor of a PC,
15 games console, tablet or mobile telephone or other computing device. The game can be implemented solely as a computer program that is stored and runs entirely on one of many processors in a remote server, and data streams or updates are supplied to the client device (e.g. tablet, smartphone etc.) to enable the client to render and display graphics and sounds; this 'web services' approach is
20 increasingly common.

Another approach is a hybrid one, in which back-end servers handle some elements of the gameplay, and for instance a Java game applet is provided to client devices and it is the locally running Java applet that generates the graphics/sounds/user interaction for gameplay on the player's client device. Some data may be fed back
25 to the back-end servers to enable scoring, interaction with other players and cross-platform synchronisation. Generally, the techniques described in this specification are not specific to any one game architecture but can be deployed on any suitable game architecture.

The game created using the techniques disclosed herein may be implemented
30 allowing a user to interact with it in different ways depending on the capabilities of the device which the user is accessing the game with. A user can interact with the game through using a touch screen where the user can select and/or move elements

on the game board with a finger or for instance with a stylus. The game can also be played with a pointing device such as a mouse or other interaction devices such as a keyboard.

5 Mobile devices may have a touch screen interface where the player can interact with the game using a finger or a pointing device such as a stylus. Some mobile devices have hard keys that complement the touch screen interface. Such hard keys may be in the form of a button or in the form of a joystick type of interaction.

10 Over the course of players playing the game, data will be produced. This data can for instance be related to a player's game performance or to game information related to a social network to which the game is connected. It is possible to gather this data, store it and make use of it for instance to improve the game. One example is by using a database to store the amount of times players try and fail a level on average. This data can then be reviewed, and if the players seem to fail a substantial amount of times before completing a level, the difficulty can be adjusted
15 accordingly. The difficulty can be adjusted through changing a score target for the level, increasing the available time or moves or giving the player for instance a booster to enhance the gameplay.

20 There can be certain performance indicators used to measure the success of the game. These indicators can for instance relate to player retention, the virality of the game and the revenue of the game.

A person skilled in the art will realise that the different approaches to implementing the game is not exhaustive, what is described herein are certain preferred embodiments. It is possible to implement the way in a number of variations without departing from the spirit or scope of the invention.

25 In a typical match-3 switcher game different game elements are swapped with each other to make moves on a game board. Figure 4, Figure 5 show exemplary implementations of a match-3 game. To gain points the player has to make moves that create matches of at least three of the same candy. If doing so, the player gains points and the matched candies are removed from the game board. As a result, new
30 candies fall into place from the top of the game board in order to fill any empty

spaces created. For all candies that are removed on the game board, points may be shown in the same colour as the candy that was removed, for example three red candies will show red points, green candies green points and so on. If a blocker element would be removed then the points shown would be in the same colour as the candies from the match that removed it.

Only swapping moves that will create at least one combination of at least three game elements of the same type are allowed.

If the player makes a match of more than 3 elements certain other new elements may appear on the game board. For instance in a typical match-3 switcher game a combination of four game elements may generate a special game element that if combined will remove one whole row or column on the game board. A combination of five game elements may generate a special game element that removes all game elements of a certain type if included in a match of game elements of that type.

A typical game mode of a match-3 switcher game provides the player with a limited number of moves to reach the level target.

Specifically in computer games and game engines, an entity is a dynamic object such as a non-player character or item. The invention can be referred to as a part of an entity system where components are used to form a whole.

The game module covers the game play of a computer game. The game module typically does not cover server communication, level progression and social network integration.

The switcher game module plays the part of the basic model for any switcher game, for example, common notions in switcher games such as switching objects every turn, matching same items, etc., are handled by the module.

Key aspects of a switcher game can be to have a board of x height and y width, elements on the board that need to be matched, matching patterns to match those elements, removal of elements when matched, element generation and randomization, game rules.

In a most basic implementation, a game development team would take the switcher module, add classes to handle drawing graphics and some data files to define their board elements and they would have a working switcher game.

5 The module receives events that trigger the game logic to be processed and fires off new events that can be used to draw the graphics. Practically, the game team would need to add game specific features to make their game unique, but having the switcher module would give a very big advantage since it would provide them with a big chunk of the code needed for free. Another advantage is that it was designed with that in mind, so there should be minimal effort in understanding it and
10 implementing the visuals and adding the game specific features.

The game module can be used for games that are designed to work stand-alone and for games that for instance require a connection to a social network.

The module may in some implementations have functions that generally are used by a switcher game. For instance, a module for a switcher game can include the
15 following functions:

- Generating a board of variable size to play the game on. It provides different generators for that purpose (random, fixed, mode-specific) and can easily be extended with more.
- Common switcher game-play logic. All switcher games need to switch
20 elements, match them, remove them when matched, add new ones to replace the removed one, shuffle when there are no moves. The switcher module handles all this.
- Common game rules. A switcher game will end under various conditions such as no moves left, score achieved, specific items collected. The module handles
25 these basic rule cases and allows them to be extended with more.
- When something goes wrong in a game, it is great if the developers have debug information (such as the board state, the switch made, etc.) in order to fix the issue. The module provides all that.
- A structure that we call entity system that allows us to create new types of
30 elements for the game by combining components. For example, if you need an element that can be matched and creates an explosion, you create a matchable and

an explosion component. If you need an element that can be matched and does not move, you reuse the matchable component and add an unmovable component. The module has a variety of such components and allows for the easy addition of new ones.

- 5 • A level editor that is used for creating levels.

Examples of specific commands that may be used in the switcher are:

- 10 • Prepare game - A command for preparing the game so that it can be played.
- 15 • Prepare board generator - A command for activating a generator for generating a playing board.
- 20 • Prepare tunnels - A command for creating tunnels.
- 25 • Handle board create failed - A command for handling cases when the board generation has failed
- 30 • Ensure no board selection - A command for ensuring nothing is selected on the board
- 35 • Set possible swaps - A command for determining the swaps that it is possible to make.
- 40 • Handle board object dragged - A command for determining what happens when an object on the board is dragged.
- 45 • Handle board object clicked - A command for determining what happens when an object on the board is clicked.
- Apply match effects - A command for executing what happens when objects are matched.
- Decrement moves until drop down item is spawned
- Update item target progress - A command for updating the target objects to match
- Add score from matches - A command for adding scores from a match to, for instance, a top list.
- Reset score multiplier - A command for resetting a multiplier for the player's score.
- Check collectible items - A command for checking for collectible items.
- Award collection score - A command for awarding a player for collecting items.

- Remove destructibles - A command for removing objects that can be destroyed.
- 5 • Apply remove effects - A command applying relevant effect after an object is removed
- Handle player move ended - A command for determining what happens when a player makes a move
- 10 • Handle shuffle failed - A command for determining what happens when a shuffle does not generate any possible moves.
- Select booster effect tile - A command for choosing where the effect a booster will happen.
- 15 • Select booster - A command for selecting a booster.
- Unselect booster - A command for de-selecting a booster that has been selected.
- 20 • Use boosters that should be used instantly - A command for using boosters that give an instantaneous effect.
- 25 • Use booster - A command for using a booster.
- Try swap items - A command for swapping items.
- End swap items - A command for triggering events supposed to happen after a swap
- 30 • Drop floating items - A command for dropping items in empty spots underneath them.
- 35 • Fill empty tiles with random items - A command for filling empty tiles in the game with randomized items.
- Remove matched items - A command for removing matched items.
- 40 • Board model - Holds information about the state of the board and the objects on it.
- Pass processor - Processes the board after a match.
- 45 • Pass terminator - Handles the condition after a pass will terminate.
- Start - Starting the pass processor.
- Make pass - Making one pass with the processor.
- 50

In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them. Figure 3 shows an example of a typical MVC software architecture.

5 A controller can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).

10 A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.

A view requests from the model the information that it needs to generate an output representation.

15 The controller module handles the user input to the device. A controller is configured to receive input signals and convert the signals into commands to be sent to the model module. The module stores the current state of the game and is configured to receive the commands and process them according to the rules of the game. The view is configured to poll the model for information related to the state
20 of the game allowing the display output to be updated.

There is usually a kind of hierarchy in the MVC pattern. The Model only knows about itself. That is, the source code of the Model has no references to either the View or Controller.

25 The View, however, knows about the Model. It will poll the Model about the state, to know what to draw. That way, the View can draw something that is based on what the Model has done. But the View knows nothing about the Controller.

The Controller knows about both the Model and the View. To take an example from a game: If you click on the "fire" button on the mouse, the Controller knows what

fire function in the Model to call. If you press the button for switching between first and third person, the Controller knows what function in the View to call to request the display change.

5 The reason to keep it this way is to minimize dependencies. No matter how the View class is modified, the Model will still work. Even if the system is moved from Windows to an app in a smart phone, the Model can be moved with no changes. However, the View and the Controller probably need to be updated.

10 Accordingly, in some embodiments, a model module can be designed for a specific game and have an interface specific to that game. The interface may be more abstract and may be a generic interface for a genre of games, for example a match-3 switcher interface. A controller can be configured to communicate with the model module through the interface. There may be a different controller module for use in each target operating system. For example, a controller module may be configured to receive input information on a mobile device running Android and another
15 controller module may be configured to receive input information from a web browser. If the interface to the model module is abstracted, the controller module may be re-used for another game in the same genre.

A library of components may be produced for use by developers. This library may be built up through the development of games to allow the re-use of components in
20 other games. For example, in a match-3 switcher game, matching three elements of a particular type may create an effect. For example, matching three elements of a type may cause the surrounding elements to be destroyed via an explosion. The mechanics of the explosion may be stored in a library to be re-used in later games or alternatively may be re-used in a different aspect of the same game. For example,
25 the explosion mechanic may then be used in connection with an item that allows a player to select a target location where the explosion may be used to destroy nearby elements. In the example, the explosion caused by the item may use different or the same graphical resources which are provided by the view module. In this way, the programmer may ensure the component when first developed is easily applicable
30 later on in development. In subsequent uses of the same component, large time savings can be achieved.

The MVC model can be used to aid multiplatform design. A library of input and output components may be readily combined to create a controller module or a view module for a particular platform. These may be re-used to allow faster development of a controller module or view module when developing a later game.

- 5 An example from the production of King's game Farm Heroes Saga will be used to showcase the strength of the system. Sometime midway during production, a new element for the game called 'Fount' was implemented. The Fount should be destroyed after a match of other elements was done next to it 3 times.
- 10 In a different architecture, a developer would need to write new logic that does all this. Without the entity system, that could take anything from 1 to 2 weeks for one developer. Having the entity system, it is possible to reuse components that were created for other types of elements. For example, the 'destructible' component with health 3 was used to implement the part when the fount is destroyed after next
- 15 matches next to it. The fact that the matches need to be next to it and not on it or on the other side of the board was taken care by setting the already existed attribute "fromOutline" to true. The whole feature was done in pretty much a day or two by adding code for the visuals and very few lines of logic code.

- The other side of this being extensible is that the module itself was made to be
- 20 reused. Any game that uses it is by definition an extension of the module. It is architected so that it makes it easy to either use the existing functionality or override parts of it at will to implement slightly or vastly different game behaviour.

- Working with badly structured code is a problem for developers. Such code is hard to understand, hard to extend and extremely expensive to maintain in the long run.
- 25 Also, it is preferable to reuse as much as possible when creating new code, so as to avoid unnecessary work. A Model View Controller based approach makes this possible.

- Another issue that most game code bases have is a distinct lack of unit tests. Unit tests are the proof that the code works as intended. In order for code to be reused
- 30 and trusted by other developers, it needs to be extensively covered with unit tests.

Games created using the invention described herein can be connected to or linked with a social network such as Facebook™ or Google+™ or a games platform with different players who can interact and see each other's progress. It is common that the users on such networks have avatars with for instance a photo of the user and/or the user's name. Such avatars can for instance also be a sign or a figure.

The social network can be located on a server that is different from the server on which the game is located, the game and the social network can also be located on the same server. In some implementations there is a direct live connection between the social network and the game platform that continuously synchronise them, in other implementations the two platforms synchronise at certain intervals, such as when the player logs into the game. The players progress when having played in offline mode (for instance completed levels and score), for instance if the player is travelling in a tunnel, can be synchronized when the player is connected to the internet.

The user and his friends' avatars can be displayed in the game or in relation to different levels in the game to show the player's progress. The avatars can also be shown in relation to indicators of the player's skill level or high score. In some implementations the avatars can be derived from a social network to which the game is connected, in other implementations they can be derived from a database related to the game. It is possible for the avatars related to users to change depending on the overall progress or performance in the game. For instance, an avatar can become larger or more visually advanced as the player plays the game for a longer time.

The user can connect with other users of the social network, either as "friends" on the social network or as "friends" within the game environment. The player can interact with other players he is connected to on the social network or who are playing the same game.

The game can be implemented to synchronize game state information and/or retrieve and connect to the social graph information and user profile of the player on a social network. It can also be connected to a proprietary network related to the game or the game developer.

The game can also be implemented so that it is connected to a plurality of social networks. The user can be given the option to select what information that can be derived and shared with which social network.

One example of how the game can be connected to a social network is the
5 Facebook™'s Open Graph API allows websites and applications to draw and share information about more objects than simply people, including photos, events, and pages, and their relationships between each other. This expands the social graph concept to more than just relationships between individuals and instead applies it to virtual non-human objects between individuals, as well. A game can typically share
10 in-game events such as that a level has been completed, that a player has passed a friend in the game or beaten a friend's high score on a level. The game can also post events, such as that a player has purchased objects in the game or received objects from other players of the game.

These social networking features can be stored in a component library. For
15 example, a component may be used to interact over a social networking API with a social networking server. The component may request a list of contacts from the server which may be provided to the user device. The list of contacts may be ordered and displayed by a component in the view module. In this example a component for fetching social networking data can be re-used in the model module
20 of any game requiring this feature. A corresponding component in the view module can be added to poll the model module for the data and produce a graphical representation on the display.

One way of implementing a game using the techniques described herein is through a web site with a plurality of casual games. This platform can be used as a basis to
25 test the performance of the game and how it is perceived by players. In some web-based implementations the game is implemented to be played in head-to-head tournaments, has a limited number of levels and no external social network connection. In some implementations the players can play the game against other players on the platform.

30 If a game proves to be successful in a web-based implementation, it can be further adapted to another type of implementation, based on a virtual terrain in which the

player progresses. This implementation typically has a connection to an external social network, and can have multiple game modes such as asynchronous and synchronous tournaments and single player mode. The nodes on the map in the game are typically different levels that the player can play.

- 5 The two implementations described above can be part of a modularised approach to developing games, which help streamline and facilitate the process of producing as well as further developing new titles.

The game can be implemented so that a player progresses through multiple levels of changing and typically increasing difficulty. Figure 6 shows an implementation of
10 the game with a virtual map layout of a game environment, displayed on the screen of the computing device used by the game player. As the player progresses through the levels in the game, his progress is represented as a journey along a path on the virtual map. Representing progress in this manner provides an additional layer of engagement for players, and also opportunities for viralisation and monetisation.

- 15 The virtual map consists of stages 1, 2 with varying number of levels 3, 4 represented by nodes on the virtual map. The user travels between levels and completes the levels one by one along a path by playing the associated game. When the player reaches the goal of a level, the next level on the path is unlocked and the player can play that level in the game. The number of stages and levels can vary
20 depending on the implementation.

In some implementations of the game, the player will be introduced to the game by tutorials explaining the fundamentals of the game. One way of doing tutorials is to force the player to make certain moves, for instance in the first level of a game the player might be prompted to make the most basic move possible without the option
25 of doing any other move. The tutorials will in most cases be concentrated to the first levels of the game, but they can also be used at later stages to explain newly introduced elements and objects.

The levels can be numbered consecutively throughout the game or they can be numbered within a stage, it is also understood that other ways of identifying the
30 stages and levels can be implemented. New stages to the virtual map 12 can be

added by the game designers at any time – so a game may be launched with say 20 levels, and after a number of weeks, there may be fifty or sixty levels present.

Stages in the game can be both locked and unlocked. In most implementations, the majority of levels start out as locked and are then unlocked as the player progresses
5 in the game. Unlocked stages can typically be replayed at any time. One way of unlocking new stages is to complete the last level on the latest stage. The user is sometimes faced with other challenges to unlock the next stage in the virtual map.

In some implementations, certain levels and stages are locked based on other criteria than the player's linear progression in the game. Such levels can for instance be
10 based on the total score the player has achieved on all levels, the average performance on levels or on the number of friends that the player has invited to play the game.

In one implementation, one challenge 7 to unlock a stage arises when traveling from one stage to another once all the levels have been completed in that stage. The
15 levels in the stage to which the player is travelling is typically locked by default, and the player must unlock them. This requires the help of for instance three friends. The player can ask friends for help by sending an in-game message within the game environment or for instance through a social network that the game is connected to. The friends can already be playing the game and do not have to be
20 'new' players, but they can be friends not already on the same social network.

The player can also pay to get instant access to the locked stage. The currency used for paying can vary between different implementations, for instance it can be hard or soft currency or it can be based on score achieved in the game. It is possible for the currency to be associated with a social network to which the game is connected,
25 or it can be associated with another platform related to the game. The player can use a combination of help from friend and payment to unlock the new stage. The cost for unlocking can in some implementations be lowered as a fraction of the total number of friends needed when help from some but not all needed friends have been received.

There can be ways of getting past a collaboration block other than asking friends for help and paying for it, which are the most common ways of passing a collaboration block. This can be done through to use of 'Mystery Quests', which gives the player the option of completing one or several challenges to unlock the block. Such
5 challenge can for instance be to play one or several past levels with modified goals in order to pass the collaboration block, for instance three levels – one for each of the locks.

These challenges are typically in the form of replaying a previously completed level but with a new goal to reach, for instance a target high score. In a typical
10 implementation, the score requirement is higher than it is for playing the level regularly, and also no other goals need to be fulfilled. For example, if the player gets to replay a level with jelly with a new target high score, the player would not need to remove the amount of jellies specified as long as the target score was reached.

15

The request for help is sent to the friend who then has the option to accept to help. The request for help can in some implementations be sent using the social network to which the game is connected; an alternative implementation is to send the request to someone external to the game (via email, text message, instant message for
20 instance) who has to join the game to respond to the help request. It can be understood that there can be variations between implementations in regards to how players respond to requests from other players. In a typical implementation, a link will be provided to the player who has been requested to help. This link can be related to a social network to the game is connected. This is one of the viralisation
25 techniques implemented in this game.

In addition to the virtual map layout in Figure 6, there can also be other levels or stages that are not part of the progress along the path in the virtual map. Such stages or levels can be present in the game associated with the virtual map at all times or can be unlocked when the user reaches a certain in-game achievement. This in-game
30 achievement can for instance be completing a specific level, reaching a predetermined high score (for instance, collecting a specific number of stars when

completing a level – highly skilled gameplay can win the user three stars) or paying virtual currency to unlock the stage or level.

The map layout in Figure 6 can be used in games connected to or linked with a social network or in a game with a user database. It is possible for users to have an account in the game or on the social network. It is common that the users on such networks have avatars with for instance a photo of the user and/or the user's name. Such avatars can also be a sign or a figure. The user's avatar is displayed on the map layout alongside the level where the user is 6. It is understood that there are different implementations of showing where the user currently is on the map. This can for instance be the latest level the user completed, the level where the player has achieved the highest score or the last completed level along the traversed path.

The user can in some embodiments be given the option to select which users should be shown on the virtual map. The users to choose from can be friends on a social network, or the user can get suggestions to show friends which meet a certain criteria, for instance friends which the player has interacted with the most in the past or friends living in the same geographic area as the player. The user can get the option to choose from other people not being friends on the social network, but that meet other certain criteria.

The user can play any of the unlocked levels on the map, so the user can go back and replay already completed levels to get a better score or beat friends' high scores.

The player is in some implementations of the game rewarded for good gameplay of a level, for instance reaching a target score or completing the level in a short time. In some implementations the user has to reach a certain number of points to complete a level, reaching this target score can be represented with a symbol such as a star. In one implementation a star is lit when the user reaches a certain number of points in a level. The user can earn more than one star on each level and the levels are re-playable to get a higher score. In some implementations the indicators representing the players' performance can be related to other goals, such as completing levels within a certain amount of tries.

The player's total number of stars collected in the game can in some embodiments unlock features. The unlocked features can for instance be power-ups, in-game currency or bonus levels. After being unlocked, such features can typically be accessed by the player in the game. Some unlockables might be given to the player while others require a purchase to be accessed.

The symbol representing how well the user has played on each level can be displayed alongside the level on the map 8, 9, 10.

In the map view, the player can hover over an unlocked level to display a thumbnail version of it. This makes it easier to find specific already completed levels, and can also give the player an idea of what to expect before actually starting a level. In a typical implementation, thumbnails cannot be displayed for levels that have not yet been unlocked. If trying to view one of these a symbol of a padlock will be in the place the miniature version of the level is supposed to be.

The thumbnail can also display how well the player has done on the level if he has played it previously. This can for instance be represented with the number of stars the player has received on that level, the actual score or some other indication.

The thumbnail can also display the player's position on the high score table in relation to the player's friends or showing what friends are on the high score table. This can be a driver for the player to replay the level to beat one of the friends.

If the game is connected to a social network or the user has connected with other players in the game, the levels can present a leaderboard showing who among the user's connections, or among a subset of the user's connections, that has the highest score. There can in some embodiments be a notification 11 shown on the map if the user that has the highest score among the friends connected to the game. Such a notification can be in the form of a message sent through for instance through the social network or an in-game message.

The type of game mode or game goals for a level can be displayed on the map as a symbol, for instance it can be a symbol for the level itself, or it can be shown in proximity to another symbol for the level. Such a symbol 3 can for instance be in

the form of an object related to the game goal, such as an hourglass representing a level with a time constraint.

In some embodiments, these features can be provided as building-block components that can drastically reduce the development time of games with this level of
5 functionality.

The MVC pattern can be further applied to server side development. A game may interact with a server to store and retrieve game progress. The server may store the user's progress information associated with multiple games or alternatively just the one game. The server may have a limited view module, for example, configured to
10 output a log and a limited controller module. The model module of a server may not vary significantly between games. Accordingly re-using server model components may aid the development of subsequent server software.

The MVC pattern can further be used to aid the debugging of code during development. The controller and view modules, which may be operating system
15 dependent, can be tested separately to the model module. The testing of the model module should operate the same regardless of the operating system and can therefore be tested individually. It may be the case that some operating systems have less sophisticated means for debugging that are more time consuming and less user friendly. For example, debugging may be easier on a desktop computer than
20 on a mobile device or a desktop computer emulating a mobile device. Accordingly, rigorous testing may be performed in the best debugging environment rather than the need to perform full testing on each intended operating system.

In some embodiments, components may be combined to create further components. For example, there may be a component which is destroyed after three nearby
25 matches and there may be a component that creates an explosion effect when destroyed. A component could easily be created which combined the two components to create an element which is destroyed after three nearby matches causing an explosion effect.

In some embodiments, components may be customisable, accepting one or more
30 parameters. For example, the component which is destroyed after three nearby

matches may be configured to be destroyed after any number of nearby matches by accepting one or more parameters upon creation.

By creating library of configurable components that can be combined in a large number of ways, creating a new game in the same genre may become a much simpler task. In an example, a first match-3 switcher may use an 8x8 grid of elements whilst a second match-3 switcher may use a 6x10 grid of elements. The model module may be instantiated for the first example with the size of the grid as 8x8. An 8x8 match-3 switcher would be produced. Alternatively, the model module may be instantiated with an argument specifying a 6x10 grid. Accordingly a new game may be instantiated with a specific configuration without the need to redevelop new software. In this way, a component can be arranged to accept one or more input parameters. These parameters may affect the way a component behaves and may be chosen by a game designer or may be randomly generated or generated according to a set of rules. These parameters allow a new game to be designed using a similar selection of components to a previous game but with a different selection of input parameters which may result in a significantly different game. Similarly, a level designer may design a level with a similar selection of components to another level but provide a different choice of parameters which may have a significant effect on how the level plays or how difficult the level may be to complete.

The initial instantiation of a game can be combined with specific game rules that can be introduced with pre-designed components to creature a unique game. These rules might set out general mechanics that are present across the whole game. In a game featuring multiple levels there may be level specific mechanics or mechanics present on some but not all levels.

The use of the MVC model can have advantages for level design in a game. In the example of a match-3 switcher, the level designer could impose rules on the types of elements present in the level. These rules could be stored in components and be reused and reconfigured. If a game is updated, new functionality from new components can be added to new or existing levels. It may be the case that two levels feature the same game elements, for example in a match-3 switcher the starting position of the elements may be the same, or the locations of various special

elements may be the same. These two levels could be made distinct through a component applying a particular rule to a level. This rule could be for example, a rule might be a specific requirement to complete a level or a change to the direction of gravity of a level. It may be the case that for a particular game, the level design
5 is entirely done by choosing the rules for a specific level.

The view module is where a significant part of the new development work may be required to make a new game according to some embodiments. The control and the model modules may be formed largely or entirely from existing components, with perhaps the need to develop some additional components. The view module of a
10 new game may require new graphical elements which may be required to distinguish a new game. Some components may be re-used, for example a component for displaying social networking information or a component for displaying elements of a match-3 switcher game.

Reference is made to Figure 7 which shows an embodiment of the present
15 application. A user device 701 is provided with an operating system 702, a graphical adapter 703, at least one input device 704 and a network interface controller 705. User device 701 is configured to run an application 706. Application 706 comprises a view module 708, a controller module 709 and a model module 707. View module 708 is configured to poll model module 707 to
20 receive game information. View module 708 is further configured to provide graphical information to graphic adapter 703 which may be configured to display game information on a display. Controller module 709 is configured to receive input information from the at least one input device 704 and determine commands to be sent to model module 707. In some embodiments, controller module 709 may
25 send commands to view module 708. Model module 707 may be configured to communicate with network interface controller 705 which may have a network connection to server 711 via network 710.

Reference is made to Figure 8 which shows a model module 801. Model module 801 contains a module 803 which incorporates the features defining the game.
30 These features include any rules or functionality that is common to all levels of the game. A plurality of modules 805, 807 and 809 correspond to a plurality of levels of the game. Each of the plurality of modules 805, 807 and 809 may contain at least

one component 811, 813 and 815. These components may comprise game rules specific to the respective level. These rules may be formed from a collection of pre-defined game rules which may be combined and configured to produce a unique level. The components 811, 813 and 815 may be selected by a game designer from
5 a library of existing components. Some of the components may be provided with one or more parameters which may affect the behaviour of functionality of the components. In this way the designer may provide a combination of components and parameters creating a new game or level from existing components.

CLAIMS

1. A software module for use in at least one computer game, configured in use to run on a processor, wherein the at least one module comprises at least one component configured to provide a game function wherein said at least one
5 component is configured to be controlled by at least one parameter the game function provided by said at least one component being determined by the at least one parameter.
2. The module of claim 1 wherein the module is configured to cause at least
10 one component to be used in one or more levels of the at least one computer game.
3. The module of any proceeding claim wherein the game function comprises at least one game element.
- 15 4. The module of claim 3 wherein said at least one parameter is configured to control at least one of functionality, behaviour and characteristics of said at least one game element.
5. The module of any preceding claim wherein the game function comprises a
20 game board.
6. The module of claim 5 wherein said at least one parameter is configured to control at least one of size and layout of the game board.
- 25 7. The method of any preceding claim wherein the game function comprises at least one game rule.
8. The method of claim 7 wherein the at least one game rules comprise at least one of:
30 switching objects;
matching objects;
removing objects; and
generating objects

9. The method as claimed in claim 8 wherein said at least one parameter is configured to control at least one of the effect of the game rules and conditions for game rules to be applied.

5 10. The module of any preceding claim configured to store debug tracking data for components of said module.

11. A method of designing at least one computer game using the module of any preceding claim.

10

12. The method of claim 11 using at least one component of said module in at least two different game levels.

13. A computing device comprising a model module, said module configured in
15 use to run on a processor, wherein the at least one module comprises at least one component configured to provide a game function wherein said at least one component is configured to be controlled by at least one parameter the game function provided by said at least one component being determined by the at least one parameter.

20

14. A computing device as claimed in claim 13, further comprising a view module

15. A computing device as claimed in claims 12 and 13, further comprising a
25 controller module.

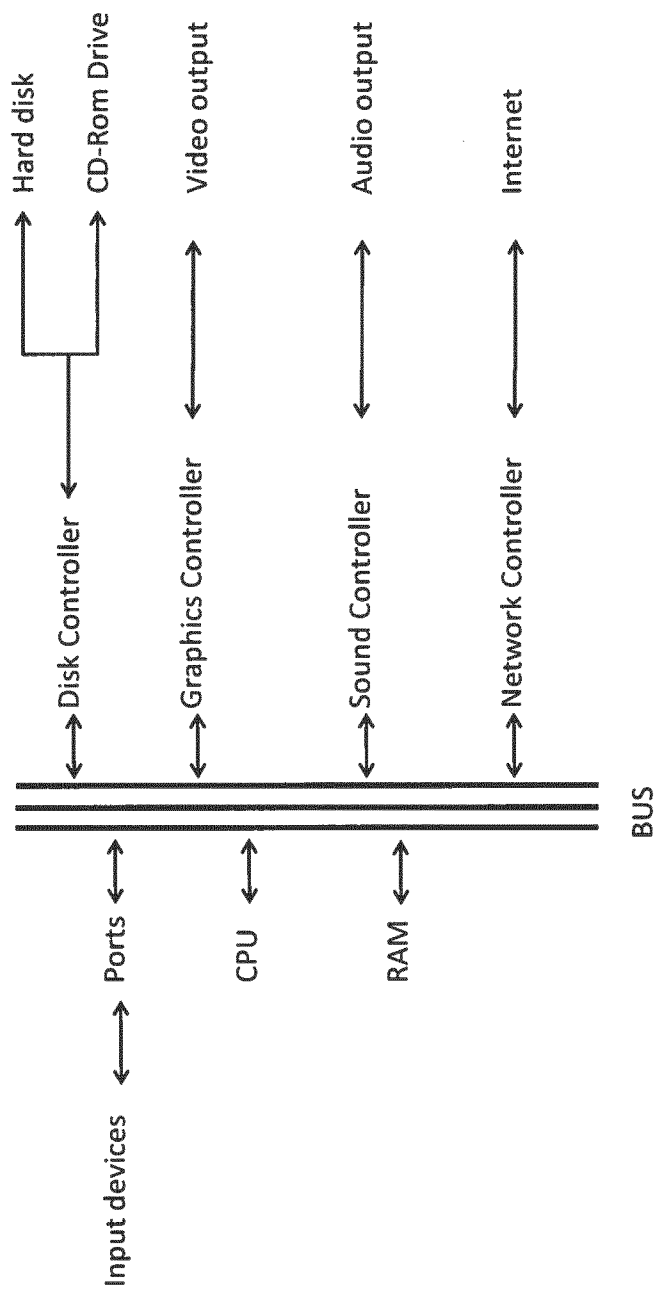


Figure 1

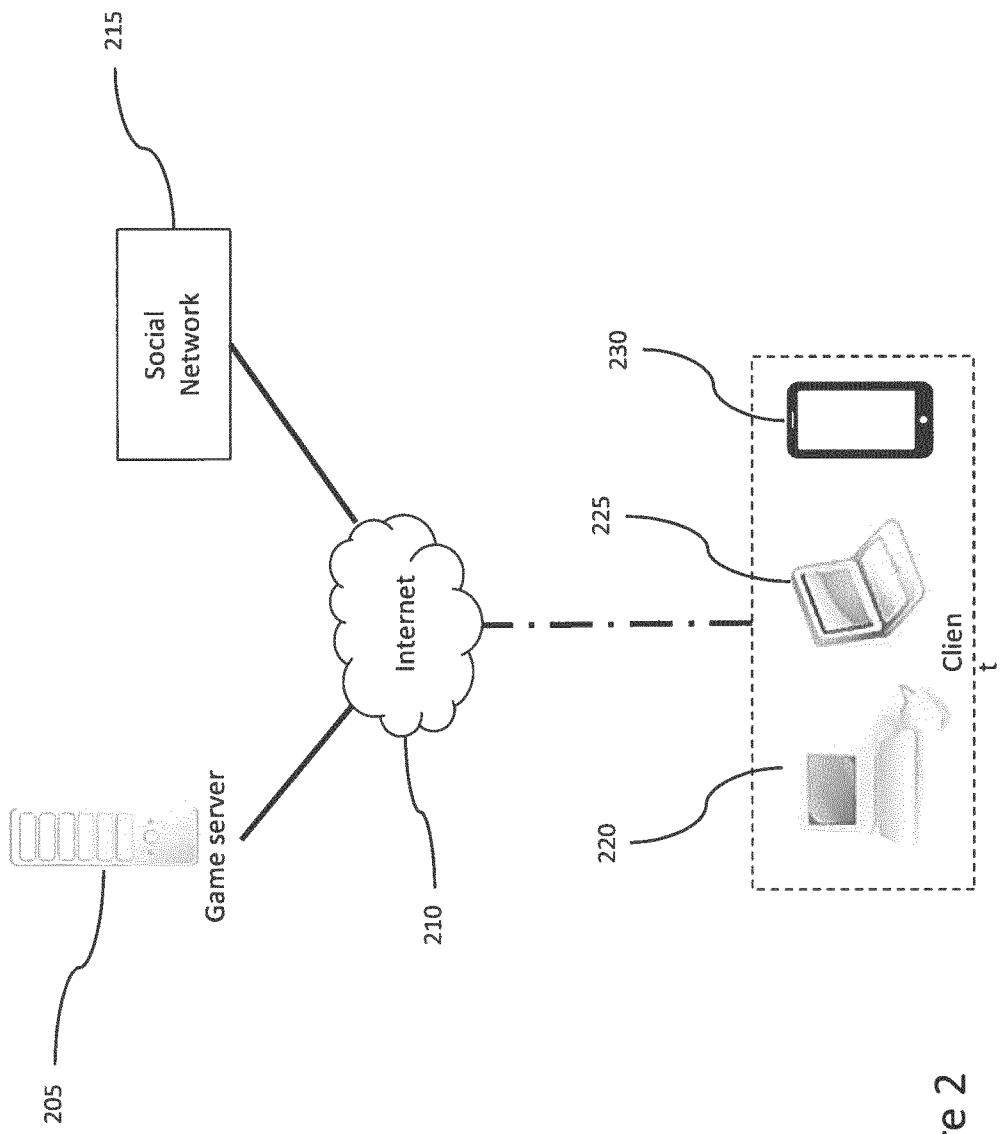


Figure 2

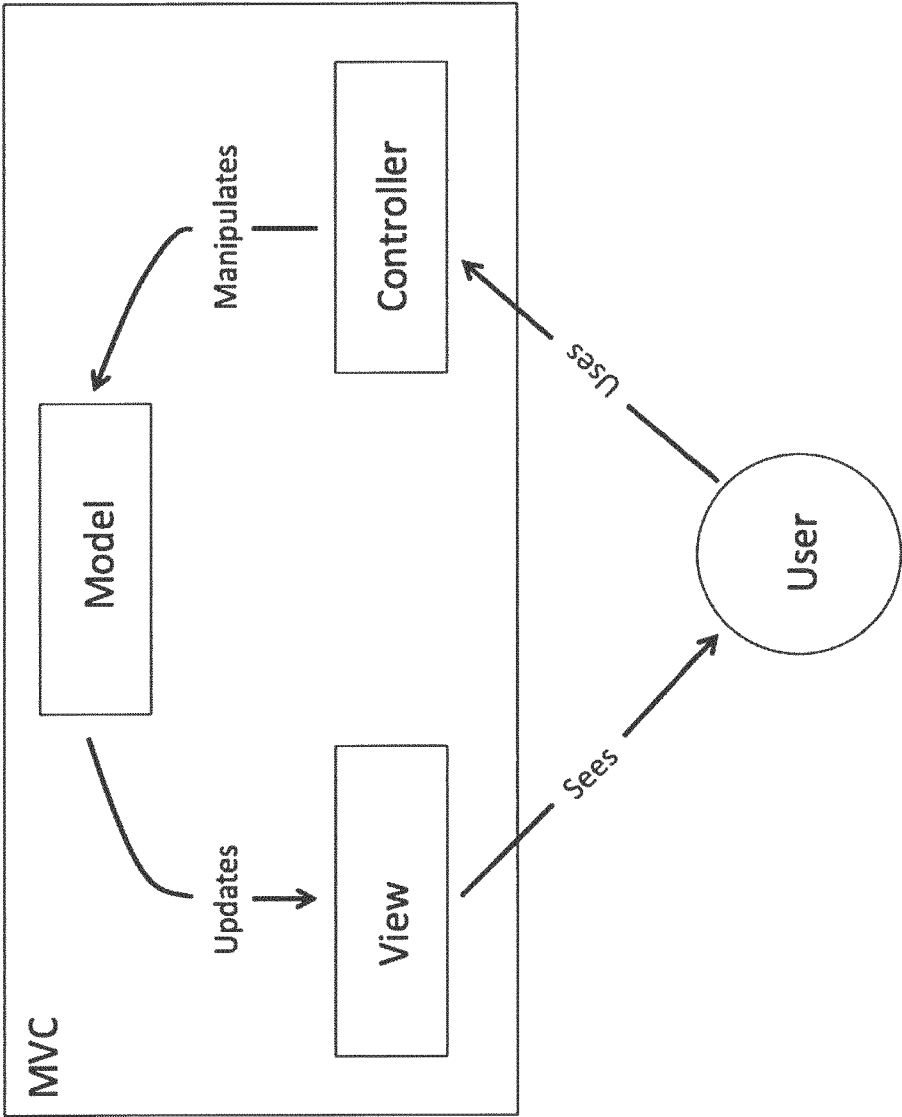


Figure 3

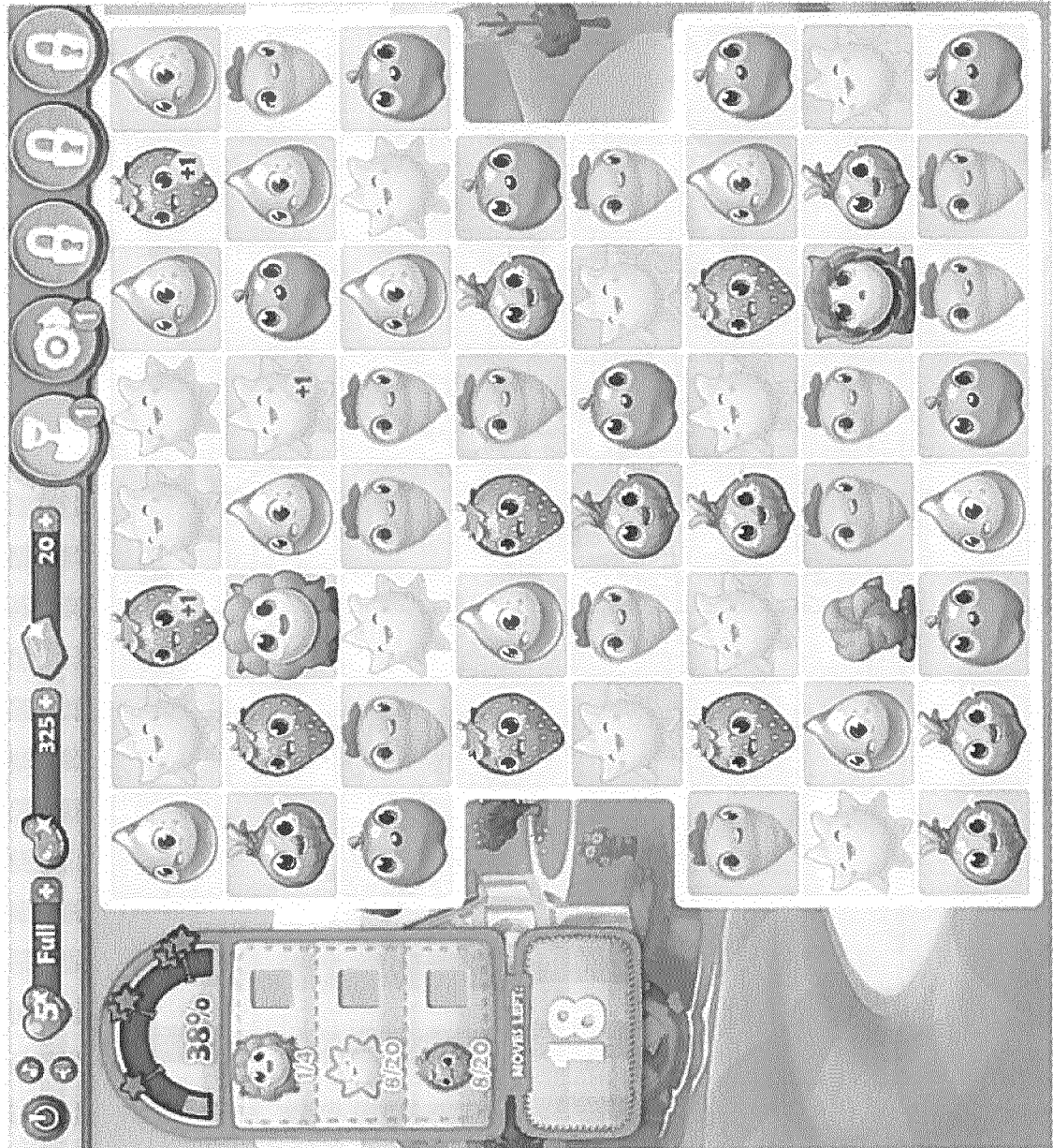


Figure 4

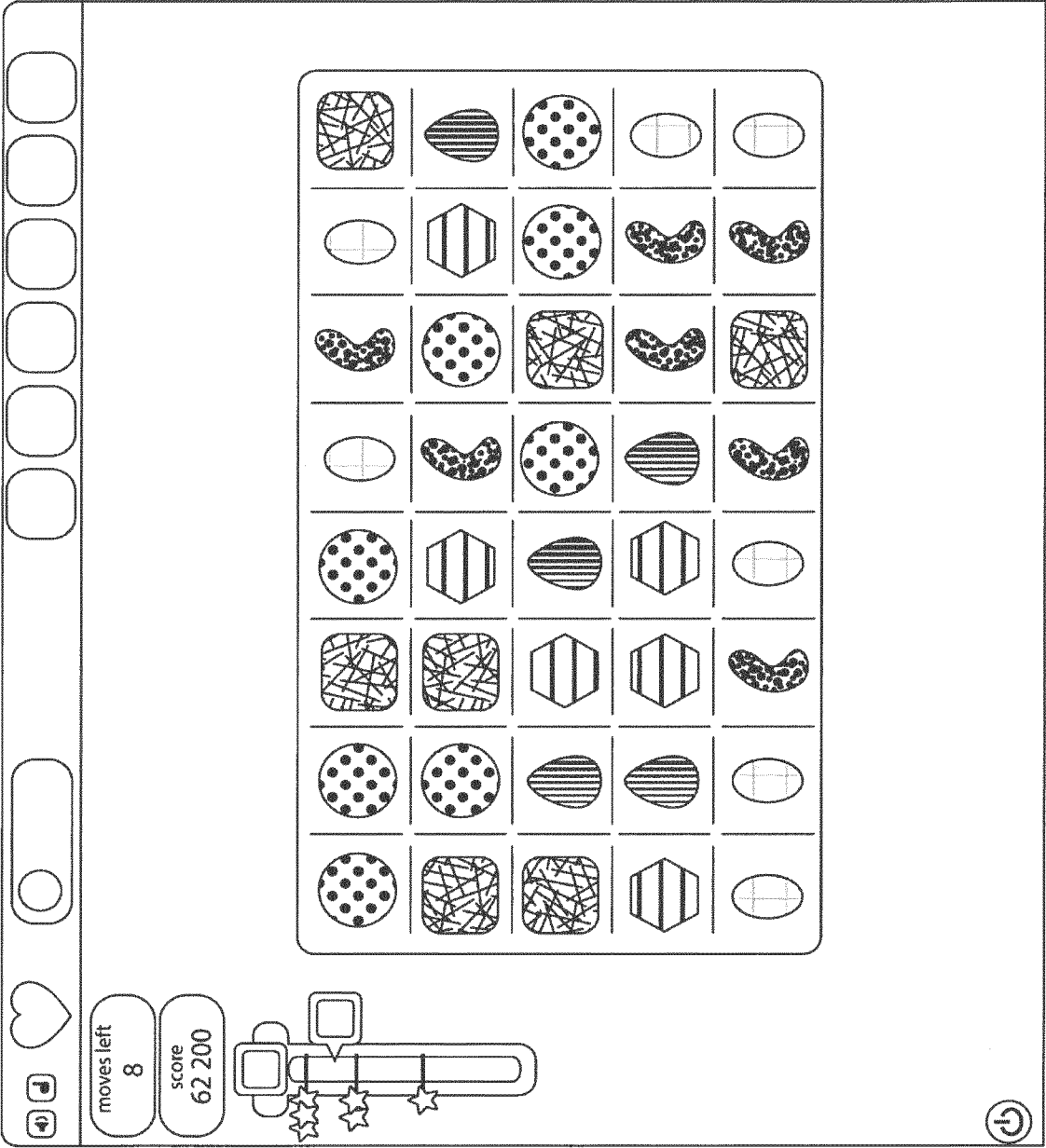


Figure 5

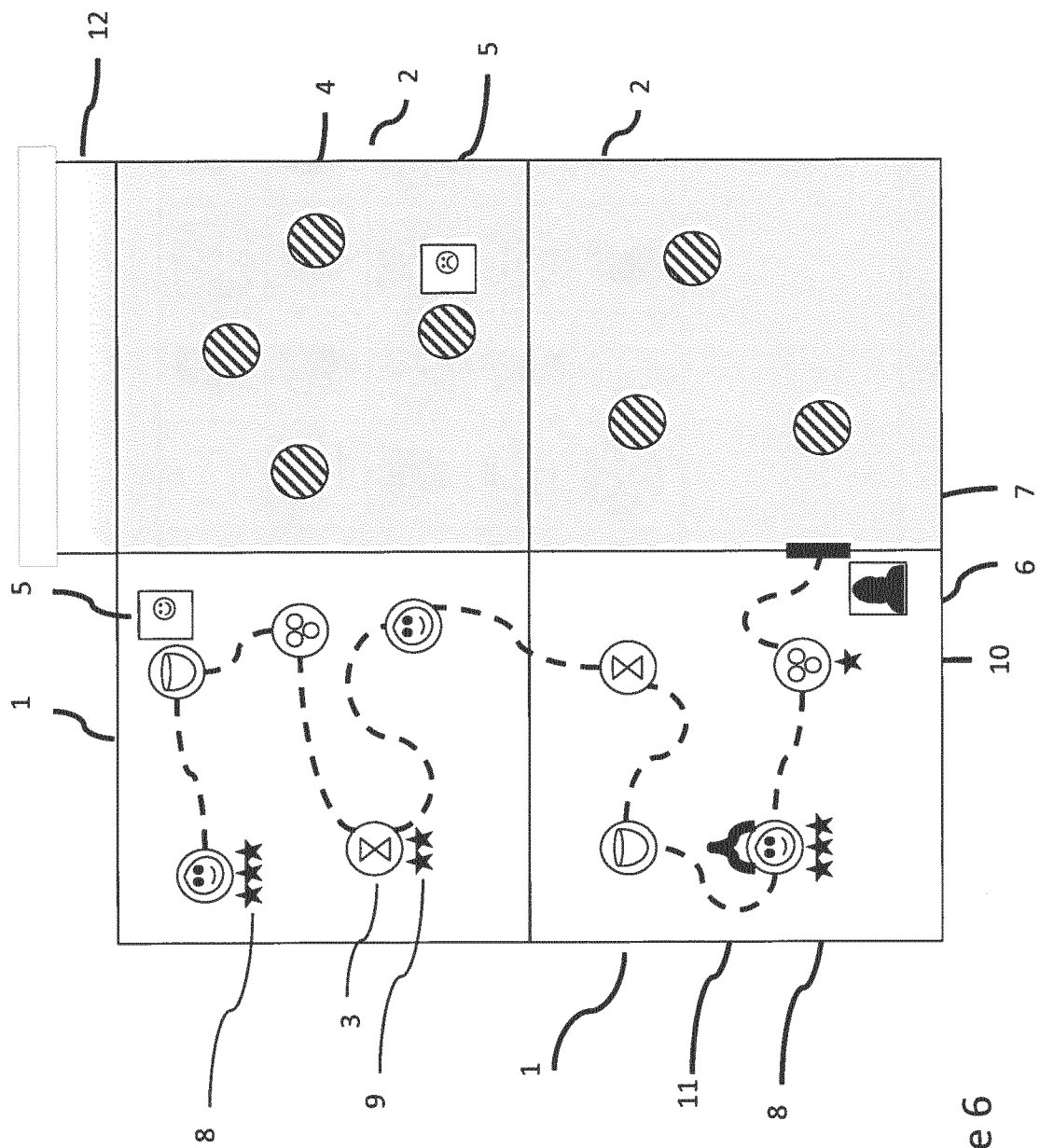


Figure 6

Figure 7

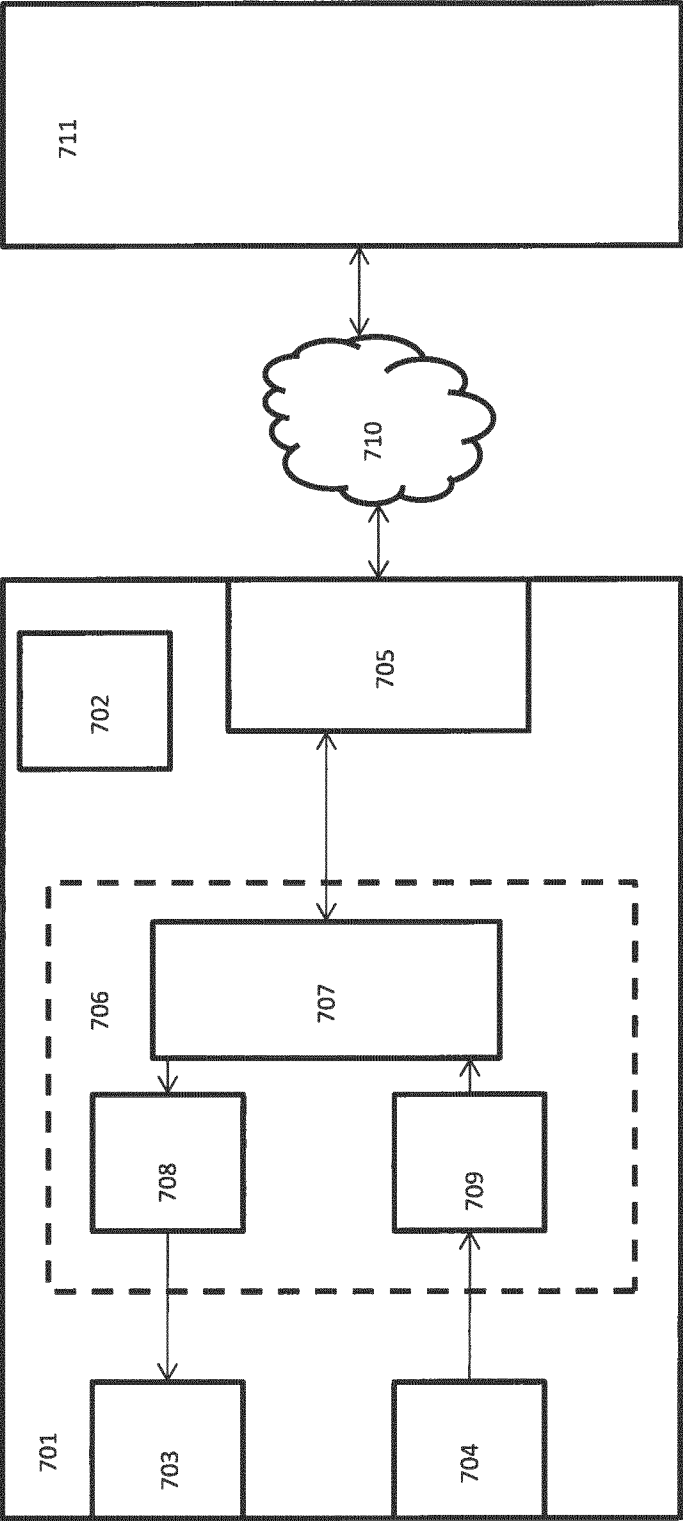
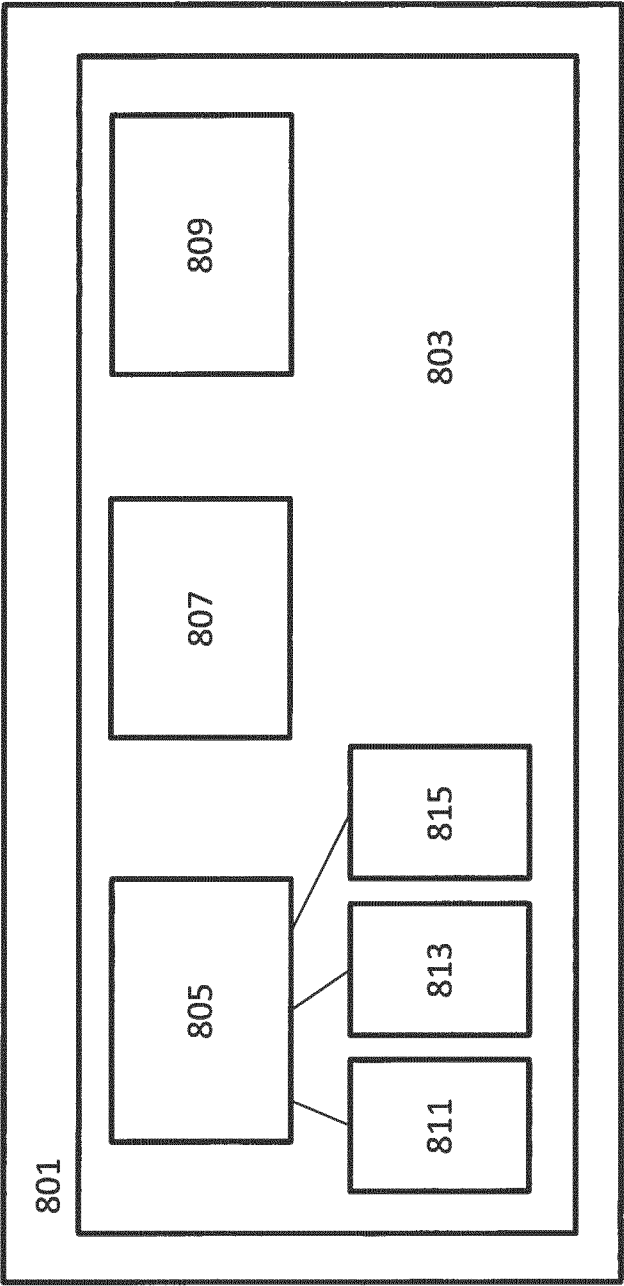


Figure 8



INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/054905

A. CLASSIFICATION OF SUBJECT MATTER
INV. G07F17/32
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G07F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/053688 A1 (HOSAKA TOSHIYUKI [JP]) 18 March 2004 (2004-03-18) paragraph [0007] - paragraph [0009]; figure 2	1-15
X	----- US 6 068 552 A (WALKER JAY S [US] ET AL) 30 May 2000 (2000-05-30) column 3 - column 7; claims 1-10	1-15
X	----- US 2006/287098 A1 (MORROW JAMES W [US] ET AL) 21 December 2006 (2006-12-21) paragraph [0014] - paragraph [0015] paragraph [0022] - paragraph [0027] paragraph [0124] - paragraph [0127] ----- -/-	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

11 April 2014

Date of mailing of the international search report

17/04/2014

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Lavin Liermo, Jesus

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/054905

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2006/039366 A2 (BALLY GAMING INC) 13 April 2006 (2006-04-13) paragraph [0025] - paragraph [0035]; claims 1-3 -----	1-15
X	US 2007/259709 A1 (KELLY BRYAN M [US] ET AL) 8 November 2007 (2007-11-08) paragraph [0106] - paragraph [0186] -----	1-15
A	Anonymous: "Why King.com's Candy Crush is crushing it on Facebook", 1 May 2012 (2012-05-01), XP055085338, Retrieved from the Internet: URL: http://blog.games.com/2012/05/01/why-king-coms-candy-crush-is-crushing-it-on-facebook/ [retrieved on 2013-10-25] page 1 - page 2 -----	1-15
A	Anonymous: "Facebook Spiel Candy Crush Saga: Cheats, Tipps, News und Videos BILDspielt", 12 April 2012 (2012-04-12), XP055085339, Retrieved from the Internet: URL: http://www.bildspielt.de/social-games/3-gewinnt/candy-crush-saga-180643/ [retrieved on 2013-10-25] page 1 - page 5 -----	1-15
A	Will Collins: "Will Collins Game Spotlight: Candy Crush Saga", 15 June 2012 (2012-06-15), XP055085345, Retrieved from the Internet: URL: https://developers.facebook.com/blog/post/2012/06/15/game-spotlight--candy-crush-saga/ [retrieved on 2013-10-25] page 1 - page 2 ----- -/--	1-15

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/054905

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	"King.com Skill Games For iPhone and Facebook: http://www.ubergizmo.com/2009/07/king-com-skill-games-for-iphone-and-facebook/", ³ 13 July 2009 (2009-07-13), XP055085351, Retrieved from the Internet: URL:http://www.google.de/imgres?client=firefox-a&hs=0gJ&sa=X&rls=org.mozilla:en-GB:official&biw=1680&bih=858&tbn=isch&tbnid=Dp0hnHVKMxtoUM:&imgrefurl=http://www.ubergizmo.com/2009/07/king-com-skill-games-for-iphone-and-facebook/&docid=fJQdwQQyyLis3M&imgurl=http://cdn.ubergizmo.com/photos/2009/7/Amazon-s [retrieved on 2013-10-25] page 1 - page 2	1-15
A	Anonymous: "Daily Bonus Level 17 Oct. 2012 Bubble Witch Saga Fan Site", ³ 17 October 2012 (2012-10-17), XP055109244, Retrieved from the Internet: URL:http://bubble-witch-saga.se/daily-bonus-level-17-oct-2012/ [retrieved on 2014-03-21] page 1 - page 2	1-15
A	Anonymous: "Bubble Witch Saga - GameSpot", ³ 31 July 2012 (2012-07-31), XP055109245, Retrieved from the Internet: URL:http://www.gamespot.com/bubble-witch-saga/ [retrieved on 2014-03-21] page 1 - page 2	1-15
A	Anonymous: "King.com expands to Google+ with Bubble Witch Saga", ³ 24 January 2012 (2012-01-24), XP055109246, Retrieved from the Internet: URL:http://www.insidesocialgames.com/2012/01/24/king-com-expands-to-google-with-bubble-witch-saga/ [retrieved on 2014-03-21] page 1 - page 2 ----- -/-	1-15

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/054905

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Anonymous: "Facebook Platform - Wikipedia, the free encyclopedia", 14 January 2013 (2013-01-14), XP055109249, Retrieved from the Internet: URL:http://en.wikipedia.org/w/index.php?title=Facebook_Platform&oldid=533105338 [retrieved on 2014-03-21] page 1 - page 2	1-15
A	Jim Squires: "Bubble Witch Saga is going mobile Gamezebo", 27 June 2012 (2012-06-27), XP055092339, Retrieved from the Internet: URL:http://www.gamezebo.com/news/2012/06/27/bubble-witch-saga-going-mobile [retrieved on 2013-12-09] page 1	1-15
A	Carter Dotson: "Bubble Witch Saga Pops Bubbles on iOS and Facebook > 148Apps > iPhone, iPad, and iPod touch App Reviews and News", 27 July 2012 (2012-07-27), XP055092342, Retrieved from the Internet: URL:http://www.148apps.com/news/bubble-witch-saga-pops-bubbles-ios-facebook/ [retrieved on 2013-12-09] page 1 - page 2	1-15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/EP2014/054905

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004053688	A1	18-03-2004	CN 1470304 A 28-01-2004
			JP 2004081815 A 18-03-2004
			US 2004053688 A1 18-03-2004
US 6068552	A	30-05-2000	US 6068552 A 30-05-2000
			US 6520856 B1 18-02-2003
US 2006287098	A1	21-12-2006	AU 2006291237 A1 22-03-2007
			CA 2622763 A1 22-03-2007
			CN 101312772 A 26-11-2008
			CN 102302855 A 04-01-2012
			EP 1933957 A2 25-06-2008
			US 2006287098 A1 21-12-2006
			US 2011294575 A1 01-12-2011
			WO 2007033005 A2 22-03-2007
WO 2006039366	A2	13-04-2006	AU 2005292080 A1 13-04-2006
			CA 2582368 A1 13-04-2006
			EP 1827632 A2 05-09-2007
			WO 2006039366 A2 13-04-2006
US 2007259709	A1	08-11-2007	US 2007259709 A1 08-11-2007
			US 2011269534 A1 03-11-2011
			US 2011269545 A1 03-11-2011
			US 2011269549 A1 03-11-2011
			US 2011269550 A1 03-11-2011
			US 2011281654 A1 17-11-2011
			US 2012004746 A1 05-01-2012
			US 2012077570 A1 29-03-2012
			US 2012088572 A1 12-04-2012
			US 2012109344 A1 03-05-2012