A method for automatic situation recognition in collaborative multiplayer Serious Games

Viktor Wendel*, Marc-André Bär, Robert Hahn, Benedict Jahn, Max Mehltretter, Stefan Göbel, Ralf Steinmetz

Technische Universität Darmstadt, Multimedia Communications Lab, Darmstadt, Germany

Abstract

One major Serious Games challenge is adaptation of game-based learning environments towards the needs of players with heterogeneous player and learner traits. For both an instructor or an algorithmic adaptation mechanism it is vital to have knowledge about the course of the game in order to be able recognize player intentions, potential problems or misunderstandings, both of the game(play) and the learning content.

The main contribution of this paper is a mechanism to recognize high-level situations in a multiplayer Serious Game. The approach presented uses criteria and situations based on the game-state, player actions and events and calculates how likely it is that players are in a certain situation. The gathered information can be used to feed an adaptation algorithm or be presented to the instructor to improve instructor decision making. In a first evaluation, the situation recognition was able to correctly recognize more than 80% of the situations in a set of game sessions.

Received on XXXX; accepted on XXXX; published on XXXX

Keywords: Serious Games, Collaborative Learning, Game Mastering, Adaptation

doi:10.4108/XX.X.X.XX

1. Motivation

Especially for game-based collaborative learning scenarios, major fields of research are adaptation of learning content, difficulty, as well as game-pace and content. First approaches to address problems in those fields have been proposed, focusing on human instructor support and Game Mastering. Many of those approaches consider what information needs to be presented to the instructor and what adaptation mechanisms are necessary and should be regarded.

A different approach is automatic adaptation of multiplayer Serious Games. Any adaptation algorithm, however, needs knowledge about the game state, the learner/player state, and player progress and behavior in order to be able to decide about proper adaptations for the present game state. In particular, an algorithm needs to be aware of problems and misunderstandings during the game. Whereas a human Game Master can rather easily recognize and judge what a player or a group of players is doing at a certain moment during the game session just by observing the scene, his/her background knowledge of the game, and human reasoning, this is extremely difficult to recognize automatically. Especially in game genres where players control an avatar in a rather open world and where they can move freely in this world, deciding for themselves about what to do next and how, it is a challenge to automatically recognize what a player - or a team - is doing at a certain moment. Examples for this are Second Life[1], mods of commercial role-playing games, or collaborative multiplayer Serious Games like Woodment [1] or Escape From Wilson Island [2].

The scenario observed here focuses on non-scene-based open world action-adventure-like games using avatars to (re-)present players. In this paper, we propose an approach for automatic situation recognition in multiplayer Serious Games. The goal is to automatically recognize what a single player or a group of players

*Corresponding author. Email: viktor.wendel@kom.tu-darmstadt.de

[1]secondlife.com/

are likely doing at a certain point in a game, based on information about their locations, their movements, their actions, and their interactions. Therefore, an interface is defined to access elementary and abstract game states, current and past player actions, game quests and (learning) tasks, and game relevant attributes. Based on this, state diagrams regarding dependencies between states are created and different kinds of criteria to calculate probabilities of situations are used. Criteria are defined based on space, time, and state, like local or global criteria, distance criteria or criteria based on game states or attributes. The concept further includes an algorithm which calculates which criteria are fulfilled using the data gathered from the game. Based on that, the algorithm calculates probabilities for players being in certain situations, like trying to solve a certain task or exploring the level, etc.

We implemented our concept as an extension of the existing collaborative multiplayer Serious Game *Escape From Wilson Island* (EFWI) which offers group tasks designed in a way such that players need to work together and communicate in order to succeed [2]. A Game Master Frontend has been implemented to enable an instructor to perform instructor tasks from inside the game at run-time [3]. The situation recognition described in this paper is a direct extension to this Game Master framework with the objective of enabling the Game Master framework to automatically recognize game relevant situation in order to inform and support the Game Master and automatically perform adaptations based on the recognized situations. An initial study to evaluate the soundness and correctness of our approach has been carried out comparing the recognized situations with the situations recognized by a real GM. The initial results are very promising. The situation recognition was able to correctly recognize the defined game situations in more than 80% of cases. However, further evaluation is required including a bigger set of users as well as additional games.

2. Related Work

2.1. Collaborative Learning

Roschelle and Teasley [4] define collaboration as " a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem". The concept of Collaborative Learning is used widely in e-learning and game-based learning. Dillenbourg [5] defined Collaborative Learning as "a situation in which two or more people learn or attempt to learn something together". Various parameters define the success of collaborative learning: One core element is the group of learners, characterized by its size and its composition. In this work, the focus is on small learner groups (2-6 players) suitable for cooperative learning scenarios. Johnson and Johnson [6] define five essential elements of cooperation which are a prerequisite for cooperation to take place in cooperative learning scenarios: Positive Interdependence, Individual Accountability and Personal Responsibility, Promotive Interaction, Appropriate Use of Social Skill, and Group Processing.

Computer-supported collaborative learning (CSCL) is the combination of computer technology and collaborative learning concepts. The fields of application are communication, coordination, cooperation in groups, and cooperative learning rooms (especially virtual learning rooms) (Haake et al. [7], p. 358). Whereas first virtual learning rooms were CSCL applications specifically designed for a CSCL purpose, most often integrating a chat system and a shared screen, later versions used existing virtual worlds like Second Live or Massively Multiplayer Online Role-Play Game (MMORPG) worlds [8].

2.2. Serious Games for Collaborative Learning

In Recent years, first CSCL Serious Games have been designed and implemented. They incorporate the CSCL principles and combine them with Serious Games principles resulting in first multiplayer Serious Games for collaborative learning [9]. Hämäläinen [10] describes an approach of a collaborative game for vocational learning focusing on design elements essential for collaboration. Reuter [11] describe an approach for designing and authoring multiplayer adventures for collaborative learning deriving concepts for puzzle design in multiplayer games.

2.3. Adaptation

The actions and tasks of the instructor can essentially be categorized as assessment and adaptation. Adaption might occur for factors like difficulty, narration, players'/learners' preferences, or the need to compensate a deficit. In general, a common definition of adaptation is " [the] ability to make appropriate responses to changed or changing circumstances" [12]. It is desirable to adapt games in various dimensions, like those stated above. Charles et al. state that "Learning and adaptation are viewed by some as a having a crucial part to play in next-generation games" [13]. Kickmeier-Rust and Albert [14] provide an overview over adaptation principles and techniques in Serious Games, like adaptive behavior of agents, motivational interventions, procedural and adaptive level and content generation. Sweetser and Wyeth [15] defined the term 'GameFlow', transferring the concept of 'Flow' [16] to games with the goal of designing and evaluating enjoyment in games. Their model includes the eight dimensions concentration, challenge, skill, control, clear goal, feedback, immersion, and social interaction. Chen [17] states that one fundamental condition for Flow in games is that the game must provide the right amount of challenges to match with the players' abilities. This, as well as the adaptation principles stated by Kickmeier-Rust and Albert [14], requires that the game knows about the current game state and in what situation players are currently in.

In order to be able to soundly adapt a game to player's needs and preferences, a sound model of a player is necessary. A widely used player model is BartleâĂŹs player model [18] for multi-user dungeons which has been adapted for roleplay games. A more generic model is proposed bei Houlette [19] which is based on a set of player traits which can be freely defined according to the game domain whereas each trait is assigned a value in the range $[0; 1]$. Smith et al. [20] developed a taxonomy of player modeling. They define a player model using the four dimensions scope of application, purpose of use, domain of modeled details, and source of modelâĂŹs derivation or motivation. The situation recognition proposed here will eventually feed and update a player model which again will be used to select appropriate adaptations.

3. Approach

3.1. Problem Statement

The approach presented here focuses on the scenario of collaborative learning in small groups using multiplayer Serious Games. The Serious Games regarded here are considered to be non-scene-based, i.e. the games are open in terms of sequence and pacing. Typical games and genres include action adventures, roleplay games, games with first person shooter mechanics, and games which rely on an open world, like many sandbox games (e.g. Minecraft). Therefore, it can be stated that it is impossible to predict the next game state from the current game state due to the high degree of player freedom. This means that it is practically impossible to foresee player movement or player actions based on the current game state. An example is a group of players standing at a certain point in a roleplay game. They can decide to go east, west, north or south, or - to make it even more complicated - they can go east for a few steps and then change direction. The possibility space is infinite merely through player movement, not including any other player actions. Thus, as each player has full control over his/her character, it cannot be predicted what will happen next. This is contrary to e.g. scene-based games where it is much easier to define concrete game-states due to limited player freedom. In that type of game, usually player actions at a specific point in the game are clearly defined and limited (i.e. by predefined dialogue options, doors to chose, or buttons to press, which deterministically lead to a next 'scene', i.e. 'state'). The player freedom in the open world is the core problem for recognizing what situation a player - or a group of players - is in. Machine-based situation recognition needs to derive a situation from certain environmental conditions which are clearly defined. This can be easy in some cases, e.g. when a player action is triggered by interacting with a game object. If a player clicks on a game object, like e.g. a berry bush, it can be stated that the player is in a 'gather berries' situation. This situation can clearly be gathered from the game as it is linked to one concrete trigger ('clicking the "Gather Berries"-button'). An example for a converse case is a situation which occurs in *Escape From Wilson Island*. Players need to surround a heron in order to hunt it. A human instructor can easily judge when players are moving to hunt and surround the heron from the players' movements. However, as there are no hard triggers (players are just moving) or events clearly defining the start of the hunting situation, this is very difficult to recognize automatically. The players' actions (including their movement) need to be evaluated algorithmically. Merely evaluating a player's position is not enough in this case as a player can be at a position for various reasons.

3.2. Game Interface

The developed situation recognition mechanism uses elemental game-data like game variables, player parameters, or elemental game actions, like moving, or triggering events by pressing a button. Based on those, game states and tasks are defined. Game situations are defined using different kinds of criteria which indicate that the situation is currently present. Finally, all situations are evaluated periodically, calculating a probability for each situation to be present at a certain point in the game session. Following, core definitions will be presented:

Definition (Game Variable). A *Game Variable* $v \in V$ is an elemental piece of information about the game. The set V contains all game variables. The function ω assigns a value of v's codomain: $\omega : v \to \{\mathbb{N}, \mathbb{R}, \mathbb{B}\}$.

Game variables can change either through game events or through player actions.

Definition (Game State). The *Game State* GS is a concrete allocation of all game variables V in the game:

$$\mathrm{GS} = \begin{pmatrix} v_0 \\ v_1 \\ \dots \\ v_{|V|} \end{pmatrix} \tag{1}$$

The game state changes whenever a game variable v ∈ V changes.

Definition (Action (repetition)). An *Action* a ∈ A is an elemental player activity which has a well-defined effect on the game state GS. A is the set of all *Actions*. The effect on GS is defined as a manipulation α of a subset of game variables $V' \subseteq V$:

$$\alpha : GS, a \rightarrow GS \; with \; GS \circ a = GS' \tag{2}$$

An action can be as simple as 'move forward' triggered by pressing the respective key, or 'gather berries' triggered by clicking on a berry bush game object. Note: Although in a game an action might have a number of prerequisites which have to be fulfilled for the action to be executable (e.g. a player needs to be within a certain distance to a game object in order to trigger the object), this is no relevant information here. The game itself takes care about checking the prerequisites. The situation recognition only needs to know when an action was performed.

Definition (Shape). A *Shape* s ∈ S describes a complex combination of game variable states. S is the set of all *Shapes*.

Thus, a shape is a boolean expression over a set of game variables which evaluates to *true* or *false*.

A shape s is considered 'active' if the boolean expression be_s describing the shape evaluates to true, else 'inactive'. A boolean expression be is either an elemental boolean expression or a composed boolean expression. An elemental boolean expression compares the current value of a game variable $v \in V$ with a target value using an arithmetic test operator. The arithmetic test operators used are: $=, \neq, >, <, \geq, \leq$; The target value is a concrete value of the respective game variable's value range (e.g. $x > 5$). A composed boolean expression combines two boolean expressions using a boolean operator. The boolean operators used to define a game state are: and, or, not, xor: The set $V_s \in V$ is the set of game variables which are used in the boolean expression be_s. The value of all used variables V_s is taken from the current game state GS.

The function f assigns {true, false} to the boolean expression be depending on the the current game state GS, i.e. depending on the current allocation of the relevant variables which are used in be. The function g assigns {active, inactive} to a shape depending on the value of its boolean expression.

$$\begin{aligned} & f : be, \mathrm{GS} \rightarrow \{true, false\} \\ & with \; f(be, \mathrm{GS}) = true \text{ if } be \text{ evaluates to true} \\ & \text{using the variable allocations in GS;} \\ \\ & g : \mathrm{S} \rightarrow \{active, inactive\} \\ & with \; g(s) = active \text{ if } f(be_s, \mathrm{GS}) = true, \text{ else } inactive; \end{aligned} \tag{3}$$

According to the above definition, at every point in the game, n states can be active simultaneously with $n \in [0; |S|]$.

Definition (Task). A *Task* $t \in$ T is defined as the tuple $t = (\mathrm{S_t}, \sigma_s, \sigma_e, \sigma_a,) \; with \; \sigma_s, \sigma_e, \sigma_a \in \Sigma, \; \delta : \Sigma, \mathrm{A} \rightarrow \Sigma$. The set of *Tasks* T contains all *Tasks*.

A task is a well-defined assignment which has to be fulfilled by one or more players. A task consists of at least two states, the start state s_s and the end state s_e. The start state defines conditions to be met before the task can be started (or for the task to be assigned). The end state is reached when the task is fulfilled. There can be a number m with $m \geq 0$ of states between start and end state. The set Σ contains all states of t. The state s_a denotes the active state. The function δ defines which successor state becomes active after performing a (player) action a. Thus, δ is a semantic note telling the situation recognition which action(s) are expected to be performed by players next for them to advance in the respective task based on the task's current state. The situation recognition can use this information to adapt the game if necessary (e.g. if players are taking too much time to execute this action, or if they lack a skill required for this action). Due to the clear unambiguous structure of a task, it is possible to use state transitions to predict players' most probable next actions and subsequently the next situation.

Whenever an action a_{now} is executed, the situation recognition checks whether a task t's active state s_{a_t} changes. If $\delta(s_{a_t}, a_{now})$ is defined, $s_{a_t} = \delta(s_{a_t}, a_{now})$, else s_{a_t}.

Definition (Region). A *Region* is a well-defined area in the game world (e.g. defined by a square or a circle).

The specific definition of a region is left to the game. The relevant information about a region is whether one or more game-relevant entities are inside an area.

In order to be able to evaluate which situation is currently present, a set of criteria is used. A situation is defined to be true if all of its criteria are fulfilled.

Definition (Criterion). A *Criterion* $c \in \mathrm{C}$, whereas C is the set of all *Criteria*, is an item which defines if a game condition is fulfilled.

For each type of criterion, an evaluator function e continuously evaluates to which extend it is fulfilled. Many criteria can either be 0 or 1 as they are evaluated in a binary way (either 'true' or 'false'). Whenever a player is referred, one or more players who are part of the situation, are meant. A player is part of a situation if he/she is part of at least one criterion of the situation.

In general, there are two types of criteria:

1. Criteria which specify one or more players.
2. Criteria which refer to one or more players.

3.3. Criteria types

Atomic Criterion

A criterion which is directly retrievable from the game (state), i.e. from a game variable, e.g. 'Is player x moving?'. An Atomic Criterion refers to the question if a game variable has a specified value. Thus, it can only be evaluated to 0 or 1, respectively 'false' or 'true' for the related condition.

$e(c) = 1$, if the respective game variable has the desired value, else 0.

The *Atomic Criterion* specifies the player which is related to the game variable, if any.

(Class) Spatial Criterion

A class of criteria which make use of player or object positions. Three concrete criterion types are defined: *Local Criterion*, *Global Criterion*, and *Distance Criterion*.

Local Criterion

A *Local Criterion* is considered fulfilled if the respective player is in the related Region. A player can either be in the specified region or not. Hence, the *Local Criterion* can only be evaluated to 0 or 1, respectively 'false' or 'true' for the related condition.

$e(c) = 1$, if the a player is in the specified area, else 0.

The *Local Criterion* specifies the player which is in the specified region.

Global Criterion

A *Global Criterion* is based on the *Local Criterion* type. It contains a set of regions and defines the relationship between them. Possible relationship types are:

1. Visiting a set of specified regions in a given order (either by one player or by several players).
2. Being in a set of specified regions at the same time (several players).

For the former relationship, the following evaluation function is used:

$e(c) = \frac{|regionsvisited|}{|regionstobevisited|}$

In this case, the Global Criterion specifies the player(s) which visited at least one of the specified areas at the correct point in time.

For the latter relationship, the criterion can be either fulfilled or not, thus:

$e(c) = 1$, if all areas are being occupied by at least one player, else 0.

In this case, the *Global Criterion* specifies the players which are in the specified areas.

Distance Criterion

A *Distance Criterion* is based on the distance between a player and another player/object/region. Therefore, a player, an object, a region and a value $max_distance$ is specified.

$e(c) = 1 - \frac{\min(current\_distance, max\_distance)}{max\_distance}$

Players are specified implicitly, here.

(Class) Temporal Criterion

A class of criteria which are based on a point in time or on a time interval. Two concrete criterion types are defined: *Time Criterion* and *Interval Criterion*.

Time Criterion

The *Time Criterion* can only be fulfilled at a certain point in time (this point can actually be a time span, like e.g. 'at night'). Therefore, two points in time are defined: t_min and t_max. Let t_now denote the current point in time:

$e(c) = 1, if t_min \leq t_now \leq t_max$

Note: t_now, t_min, and t_max can refer to continuous time scale (global time) or recurring time (e.g. time of day). If it refers to a continuous time scale, there can only be one time interval where $e(c) = 1$. Otherwise, there can be a time span with $e(c) = 1$ every cycle. There is no player specified or referred to.

Interval Criterion

The *Interval Criterion* is used to measure the time distance since the last occurrence of an event or action. Therefore, an action or an event is specified, as well as a $max\_distance$ value. Let t_now denote the current point in time, and $t_lastoccurence$ the point in time when the action or event occurred the last time:

$e(c) = \frac{\min((t_now - t_lastoccurence), 1)}{max_distance}$.

A player is specified if the Interval Criterion refers to a player action. The player who triggered that action last is the specified player.

	Explanation of field	Value
Name	Name of the situation	String
Description	Description of the Situation	String
Caused by	'Player' or 'Group'	String
$Criterion_0$ ... $Criterion_n$	Criteria which need to be fulfilled for this situation to be true	$C_{sit} \subseteq C$

Table 1. Situation

(Class) State-oriented Criterion

The class of state-oriented criteria is based on a certain (part of the) game state, i.e. a shape or a player attribute. Three concrete criteria are defined: *Shape Criterion*, *Attribute Criterion*, and *Inventory Criterion*.

Shape Criterion

Uses a concrete shape, i.e. is considered fulfilled if the respective shape is active:

$e(c) = 1$, if the a shape is 'active', else 0.

There is no player specified or referred to.

Attribute Criterion

The *Attribute Criterion* is based on player attributes. If a player attribute has a certain value, is above or below a certain threshold, or is in a certain range, the criterion is considered fulfilled. Therefore, either x_{min} is specified as a lower threshold, x_{max} is specified as an upper threshold, both x_{min} and x_{max} are specified as a range, or x_{eq} is specified as a concrete value. Let x_{now} be the current value of the player attribute.

1. If only x_{min} is specified: $e(c) = 1$, if $x_{now} \geq x_{min}$, else 0.
2. If only x_{max} is specified: $e(c) = 1$, if $x_{now} \leq x_{max}$, else 0.
3. If x_{min} and x_{max} are specified: $e(c) = 1$, if $x_{min} \leq x_{now} \leq x_{max}$, else 0.
4. If only x_{eq} is specified: $e(c) = 1$, if $x_{now} = x_{eq}$, else 0.

The player whose attribute is evaluated, is the specified player.

Inventory Criterion

This criterion is used to model the necessity of an item to be in a player's inventory.

$e(c) = 1$, if the specified item is in the player's inventory, else 0.

The *Inventory Criterion* refers to a player which is part of another criterion of the situation.

Task Criterion

This criterion is used to make a task a prerequisite of a situation. The task needs to be either in the state 'not started', 'ongoing', or 'finished'. Therefore, x_{task} is specified as 'not started', 'ongoing', or 'finished'. Let x_{now} be the current state of the task.

$e(c) = 1$, if $x_task = x_{now}$, else 0.

The *Task Criterion* does not specify or refer to a player.

3.4. Situation Evaluation

Definition (Situation). A *Situation sit* is a point of interest in the game which can be considered interesting or relevant due to its meaning for the game or game purpose.

In contrast to shapes, situations are not easily tangible by defining a boolean expression over a set of concrete game variables. Rather, situations are used to describe vague incidences during the course of the game. Those incidents would usually be categorized by a human person which is able to recognize and judge player behavior and game situations.

A situation is defined via a set of criteria $\mathrm{C_{sit}}$: $sit := \mathrm{C_{sit}} \subseteq \mathrm{C}$ A situation is considered partially present if only a part of its criteria is fulfilled or one or more of its criteria are only partially fulfilled. A situation is either caused by one player or by the whole group.

Note: in contrast to a shape, which has two concrete states: *active* and *inactive*, a situation has a continuous value range of $[0;1]$ indicating the probability that the situation is present.

For each situation, the situation recognition calculates how likely it is that the situation is currently present. This is performed on a cyclic basis, like every frame. However, for performance reasons this can be reduced to once per second. Therefore, it evaluates the situations' criteria. For each criterion, an evaluator function e assigns a value between $[0;1]$: $e : \mathrm{C} \longrightarrow [0;1]$ The situation's sit degree of presence $e(sit)$ thus is:

Hence, it is possible to assign a probability value (between 0 and 1) for each defined situation in the game. The evaluator then orders all situations according to their evaluation value in a descending order, trimming those situations where $e(sit) < \delta$ with δ being a threshold to define a minimum probability.

$$e(sit) = \frac{\sum_{i=0}^{|C_{sit}|-1} e(c_i)}{|C_{sit}|} \quad (4)$$

3.5. Situation Examples

Following, two examples will be presented showing the use of criteria to define a situation:

Example 1: Skyrim - Buying a house in the city of Whiterun:

- Local Criterion: Is the player in Whiterun?
- Task Criterion: Is the player 'Thane' of Whiterun?[2]
- State Criterion: Did the player already slay the first dragon?
- Distance Criterion: Is the player close to the Non-Player-Character Proventus Avenicci?
- Inventory Criterion: Does the player possess more than 5000 pieces of gold?

If all of the criteria listed above are fulfilled, there is a high probability that the player is on his/her way to buy a house in Whiterun. If however, only some of the criteria are fulfilled, the player however is not Thane, it is still possible that he/she is trying to buy the house, but just does not yet know that he needs to be Thane. Or, he/she approaches the NPC because he/she wants to buy something else. Subsequently, the probability that the player is about to buy the house is still relatively high.

Example 2: EFWI - Building the log hut:

- Local Criterion: Are the players close to the hut building area?
- State Criterion: Is the first part of the hut build already?
- Atomic Criterion: Is a large palm being carried?

If all of those criteria are fulfilled, it can be assumed with a very high probability that players are trying to finish building the log hut. If, however, only a part of those criteria is fulfilled, the players are probably trying something else (like building the raft, or carrying logs to make firewood) or they, for example, do not know where they have to build the hut. Therefore, the probability that the players are actually building the log hut is lower, but significantly above zero.

[2]The player can become Thane of Whiterun (a title of nobility) by solving a quest, i.e. a task

4. Implementation

We implemented our approach as a Unity3d-based library coded in C# and used the existing Serious Game Escape From Wilson Island for a first evaluation. Due to its Game Mastering interface, EFWI already provides access to its Game Variables and Actions. That provides all necessary information for the situation evaluation.

4.1. Escape From Wilson Island

EFWI is a multiplayer Serious Game focusing on collaboration and teamwork. The narration can be described as a collaborative 'Robinson Crusoe scenario'. The tasks to be solved by the players are designed in a way such that players need to work together and to coordinate their actions. The overall task is to escape from a deserted island where the players stranded. In order to achieve this goal, the players need to build a log hut for shelter, establish a food supply, build a raft, find and fill a gas bottle, steer the raft towards the second island, and ignite a signal fire. The single tasks require a high amount of teamwork and coordination. Carrying a palm to build the log hut, for example, requires three players to coordinate their movement to not let the palm fall down. More detailed information about the game can be found in [2]. The Game Master concept and frontend is described in [3].

4.2. EFWI Tasks

In EFWI, players need to solve several sub-tasks, like building the log hut, or filling the gas bottle, in order to finally fulfill the overall task of escaping from the island.

4.3. EFWI Example Situation

The following example describes the definition and evaluation of the Situation 'Filling Gas Bottle'. Table 2 shows a list of the defined Criteria.

The gas bottle can only be filled at night. Therefore the first Criterion evaluates to 0 if it is day, and to a value between 0 and 0.2 at night, with 0.2 at midnight. A player needs to provide light to find the correct geyser at night. Therefore, a player needs to have a flashlight in his/her inventory. If this is the case, the second Criterion is evaluated to 0.2, otherwise to 0. Players also need to have the empty bottle in their inventory. So this criterion evaluates to 0.2 or 0, if not. Players can only fill the bottle, if they are in the 'Geyser Region' of the island. If players are close to the geysers, this Criterion evaluates to 0.2. The farther away players are, the lower this value becomes, until it reaches 0 when players are far enough away. Finally, the flashlight needs to be on, so that players can fill the bottle. So this Criterion evaluates to 0.2 if the flashlight is on, and to 0 if not. Thus, if all of those Criteria are fulfilled completely,

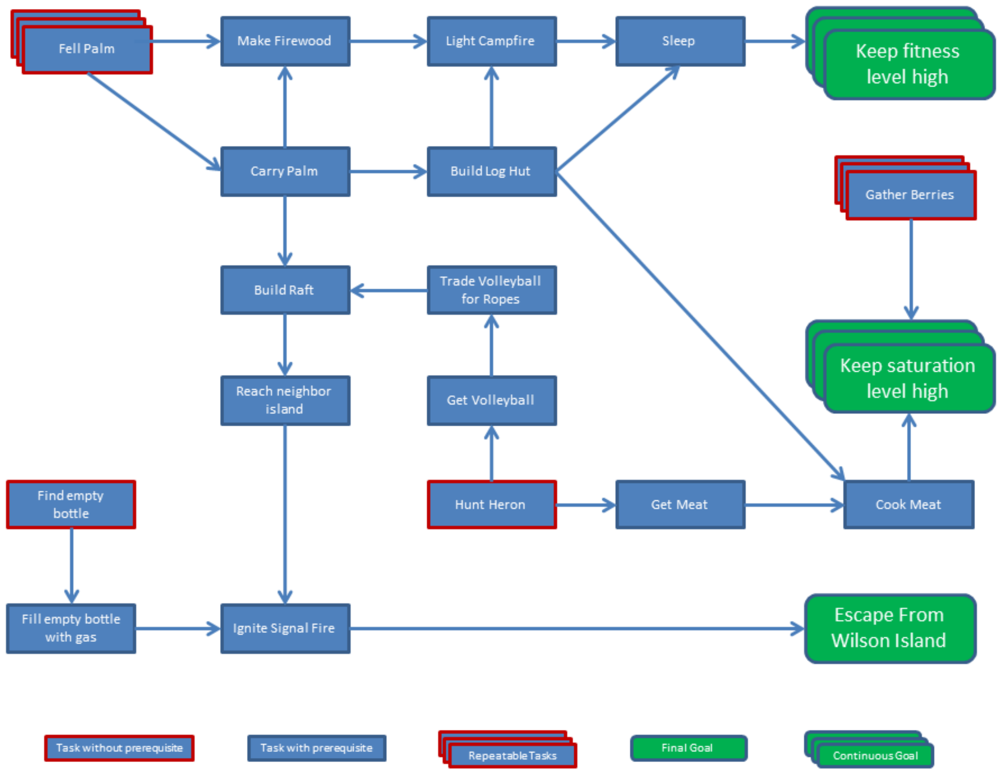


Figure 1. Schematic of all Escape From Wilson Island tasks

Criterion Name	Criterion Type	Explanation	Value Range
DayNight	TimeCriteria	Day-night-Cycle	[0;0.2]
FlashlightInInventory	InventoryCriteria	Flashlight in inventory?	[0;0.2]
EmptyBottleInInventory	InventoryCriteria	Empty Bottle in inventory?	[0;0.2]
GeyserRegion	LocalCriteria	Player in geyser Region?	[0;0.2]
FlashlightUsed	StateCriteria	Flashlight being used?	[0;0.2]
FilledBottleInInventory	InventoryCriteria	Full Bottle in inventory?	[0;-1]

Table 2. Criteria of the' Filling Gas Bottle' situation

the Situation would be evaluated to a value close to 1. Once players filled the bottle with gas, they won't try to fill it again. Therefore, the FilledBottleInInventory Criterion is added which is evaluated to -1 if a player already has a filled gas bottle, zeroing out all positive evaluations of the other Criteria. So, even if players are moving around the geysers at night, having a flashlight on, they probably are not trying to fill the bottle, if it is already filled.

The following Table 3 contains *Game Variables* and *Actions* which are being accessed to evaluate the *Criteria*:

4.4. EFWI Situation Recognition GUI

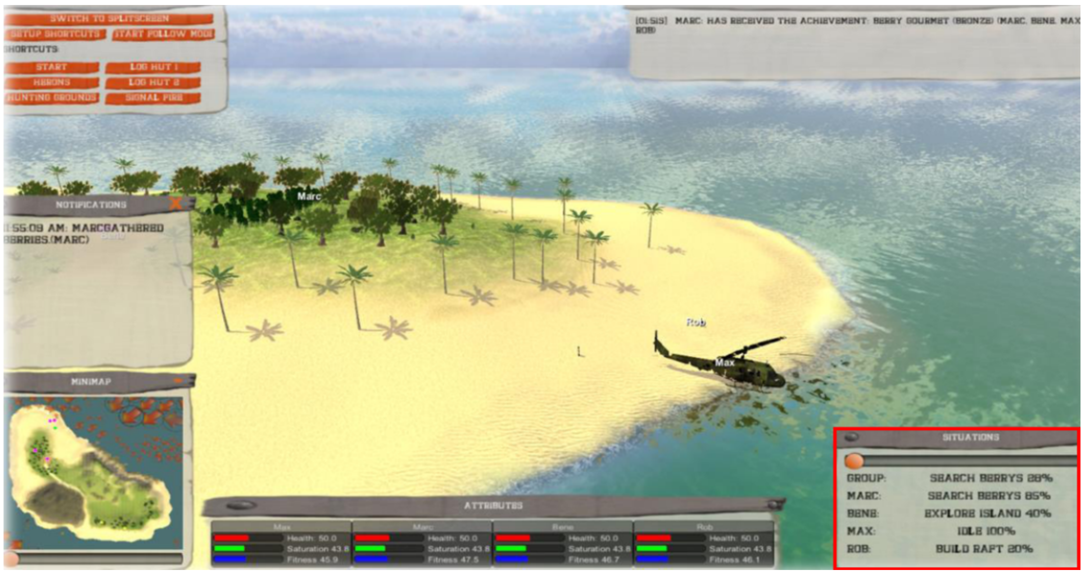
In the Game Master GUI (Figure 2), the highest evaluated situations are being displayed for each player and for the whole group. The GM can adjust how many situations to display for each player/group between only the most significant or all situations.

5. Evaluation

An initial study to evaluate the soundness and correctness of our approach has been carried out comparing the recognized situations with the situations recognized by a real GM. Therefore, the following situations have been defined as relevant: 'Build log hut', 'Build raft', 'Drive raft', 'Explore island', 'Fill bottle', 'Hunt heron', 'Idle', 'Ignite fire', and 'Search berries'. Those situations cover the first part of the game until steering the raft towards the second island.

	Name	Type
Game Variables	GameTime	Float
	FlashlightInInventory	Bool
	EmptyBottleInInventory	Bool
	FullBottleInInventory	Bool
	Player Position (player attribute)	(float,float,float)
Player Actions	PickupBottle	
	Fill Bottle	
	Move	
	SwitchFlashlightOnOff	
Regions	GeyserArea	

Table 3. Example Game Variables, Player Actions, and Regions in EFWI



[01:51S] MARC: HAS RECEIVED THE ACHIEVEMENT: BERRY GOURMET (BRONZE) (MARC, BENE, MAX, ROB)

1:55:09 AM: MARCGATHERED BERRIES.(MARC)

Max	Marc	Bene	Rob
Health: 50.0	Health: 50.0	Health: 50.0	Health: 50.0
Saturation 43.8	Saturation 43.8	Saturation 43.8	Saturation 43.8
Fitness 45.9	Fitness 47.5	Fitness 46.7	Fitness 46.1

GROUP:	SEARCH BERRYS 28%
MARC:	SEARCH BERRYS 85%
BENE:	EXPLORE ISLAND 40%
MAX:	IDLE 100%
ROB:	BUILD RAFT 20%

Figure 2. Game Master overview over the game including the situation recognition

5.1. Setup

Five game sessions (with two runs each) were played with random players. In the first run, a Game Master was watching the gaming session via the Game Master Frontend. The visualization of recognized situations was disabled. The Game Master was instructed to write down what he/she thought what a player/the group was doing once per minute or whenever he/she thought players were doing something new. The notes were provided with a timestamp. Those notes were then compared to the situations recognized by the situation recognition. The situation recognition was always able to recognize the same situation for the group within 1-5 seconds of the actual situation happening. For single players, the correct situation was always among the three most significant evaluated situations. After that, in a second game run with the same Game Master the visualization of the situation was enabled. Now, GMs again observed the game. After the game run, the GMs gave feedback on the situations which the situation recognition had proposed. All of the GMs stated that the situation they thought to be present was among the first three situations recognized for a player, and among the first two situations for the group.

5.2. Discussion

In sum, the initial study showed that the situation recognition worked very well for the game EFWI. A Game Master can use the information displayed as a means to reduce to cognitive load during the process of moderating a game session. The additional information might help to judge the current state of the game and the learning/gaming process and help to decide about adaptation measures. In terms of automatic adaptation of multiplayer games, it can be stated that the automatic recognition of game situations can be viewed as a prerequisite of being able to make sound adaptations to a game. Therefore, the situation recognition presented here might be a first step towards a sound adaptation of non-scene-based multiplayer games with a high amount of player freedom. Being able to automatically recognize the situation, a group of players/learners is in, might help to recognize problems in the learning/gaming process and subsequently react with an appropriate game adaptation. However, a comprehensive study under laboratory conditions is required for further, more precise statements. Moreover, it is necessary to evaluate the situation recognition framework in other similar games for evidence about the genericity of the concept.

6. Conclusions

In this paper we presented an approach for automatic situation recognition in collaborative multiplayer Serious Games as a foundation for automatic adaptation of collaborative multiplayer Serious Games. Our approach uses basic information about the game, like elemental game variables and player parameters as well as player actions and game events. Situations are defined based on criteria which describe a game situation using elemental game information. An evaluation algorithm periodically evaluates the probability of a situation to be present. We implemented our approach as an extension of the existing collaborative multiplayer Serious Game *Escape From Wilson Island* and performed an initial evaluation. First results showed that our approach is able to correctly recognize the defined situations in congruence with human instructors. However, a more comprehensive study is required for more detailed results as well as information about the transferability of our concept to other games of similar type.

References

[1] Wendel V, Babarinow M, Hörl T, Kolmogorov S, Göbel S, Steinmetz R. Woodment: Web-Based Collaborative Multiplayer Serious Game. In: Pan Z, Cheok AD, Müller W, Zhang X, Wong K, editors. Transactions on Edutainment IV. vol. 6250 of Lecture Notes in Computer Science. 1st ed. Springer; 2010. p. 68–78.

[2] Wendel V, Gutjahr M, Göbel S, Steinmetz R. Designing Collaborative Multiplayer Serious Games. Education and Information Technologies. 2013;18(2):287–308.

[3] Wendel V, Göbel S, Steinmetz R. Game Mastering in Collaborative Multiplayer Serious Games. In: Göbel S, Müller W, Urban B, Wiemeyer J, editors. E-Learning and Games for Training, Education, Health and Sports - LNCS. vol. 7516. Darmstadt, Germany: Springer; 2012. p. 23–34.

[4] Roschelle J, Teasley S. The Construction of Shared Knowledge in Collaborative Problem Solving. In: OŠMalley C, editor. Computer-supported Collaborative Learning. Berlin: Springer-Verlag; 1995. p. 69Ű–97.

[5] Dillenbourg P. What Do You Mean by Collaborative Learning? In: Dillenbourg P, editor. Collaborative-learning: Cognitive and Computational Approaches. Oxford: Elsevier; 1999. p. 1–19.

[6] Johnson DW, Johnson RT. Making cooperative learning work. Theory into practice. 1999;38(2):67–73.

[7] Haake J, Schwabe G, Wessner M. CSCL-Kompendium: Lehr-und Handbuch zum computerunterstützten kooperativen Lernen. Oldenbourg Wissenschaftsverlag; 2004.

[8] Eustace K, Lee M, Fellows G, Bytheway A, Irving L. The Application of Massively Multiplayer Online Role Playing Games to Collaborative Learning and Teaching Practice in Schools. In: Atkinson R, McBeath C, Jonas-Dwyer D, Phillips R, editors. Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference; 2004. .

[9] Zea NP, Sánchez JLG, Gutiérrez FL, Cabrera MJ, Paderewski P. Design of Educational Multiplayer Videogames: A Vision From Collaborative Learning. Advances in Engineering Software. 2009;40(12):1251–1260.

[10] Hämäläinen R. Using a game environment to foster collaborative learning: a design-based study. Technology, Pedagogy and Education. 2011;20(1):61–78.

[11] Reuter C, Wendel V, Göbel S, Steinmetz R. Multiplayer Adventures for Collaborative Learning With Serious Games. In: Felicia P, editor. 6th European Conference on Games Based Learning. Reading, UK: Academic Conferences Limited; 2012. p. 416–423.

[12] Kaukoranta T, Smed J, Hakonen H, Rabin S. Understanding pattern recognition methods. AI game programming wisdom. 2003;2:579–589.

[13] Charles D, Kerr A, McNeill M, McAlister M, Black M, Kcklich J, et al. Player-centred game design: Player modelling and adaptive digital games. In: Proceedings of the Digital Games Research Conference. vol. 285. Citeseer; 2005. .

[14] Kickmeier-Rust MD, Albert D. Educationally Adaptive: Balancing Serious Games. Int J Comp Sci Sport. 2012;11(1).

[15] Sweetser P, Wyeth P. GameFlow : A Model for Evaluating Player Enjoyment in Games. Computers in Entertainment (CIE). 2005;3(3):1–24.

[16] Csikszentmihalyi M. Flow: The Psychology of Optimal Experience. Harper Perennial; 1991.

[17] Chen J. Flow in games (and everything else). Communications of the ACM. 2007;50(4):31–34.

[18] Bartle R. Hearts, clubs, diamonds, spades: Players who suit MUDs. Journal of Virtual Environments. 1996;1(1):19.

[19] Houlette R. Player Modelling for Adaptive Games. AI Game Programming Wisdom II. 2004;p. 557–566.

[20] Smith AM, Lewis C, Hullet K, Sullivan A. An inclusive view of player modeling. In: Proceedings of the 6th International Conference on Foundations of Digital Games. ACM; 2011. p. 301–303.