# Game Character Ontology (GCO)

## A Vocabulary for Extracting and Describing Game Character Information from Web Content

Owen Sacco
Institute of Digital Games,
University of Malta
http://www.game.edu.mt
owensacco@gmail.com

Antonios Liapis
Institute of Digital Games,
University of Malta
http://www.game.edu.mt
antonios.liapis@um.edu.mt

Georgios N. Yannakakis
Institute of Digital Games,
University of Malta
http://www.game.edu.mt
georgios.yannakakis@um.edu.mt

## ABSTRACT

Creating video games that are market competent costs in time, effort and resources which often cannot be afforded by small-medium enterprises, especially by independent game development studios. As most of the tasks involved in developing games are labour and creativity intensive, our vision is to reduce software development effort and enhance design creativity by automatically generating novel and semantically-enriched content for games from Web sources. In particular, this paper presents a vocabulary that defines detailed properties used for describing video game characters information extracted from sources such as fansites to create game character models. These character models could then be reused or merged to create new unconventional game characters.

## CCS CONCEPTS

• **Information systems** → **Ontologies**; • **Theory of computation** → *Action semantics*;

## KEYWORDS

Vocabularies, Ontologies, Semantic Web, PCG

**ACM Reference format:**
Owen Sacco, Antonios Liapis, and Georgios N. Yannakakis. 2017. Game Character Ontology (GCO). In *Proceedings of Semantics2017, Amsterdam, Netherlands, September 11–14, 2017,* 8 pages.
DOI: 10.1145/3132218.3132233

## 1 INTRODUCTION

The Web contains vast sources of content that could be reused to reduce the development time and effort to create games. Our vision [15] is to generate novel and semantically-enriched content for games from diverse Web sources. In particular, we envision a game character generator that extracts character information directly from the Web such as from wiki articles or images that are freely available and generate new characters from already existing ones. Following a semantic-based game generation approach not only can reduce the time and cost of game content creation but also directly contribute to web-informed yet unconventional game design.


*Semantics2017, Amsterdam, Netherlands*

DOI: 10.1145/3132218.3132233

Games are composed of different domains (or *facets*) that contribute to the game's look, feel and experience [10]. These facets include visuals, audio, narrative, gameplay, game design and level design. Each facet can be regarded as an independent model containing specific content, and a game is created when each of these models are interlinked together based on the game's requirements. Current work on automatic generation of content comprise of algorithms that generate limited in-game entities, such as *SpeedTree* [18] that generates trees and vegetation as part of the visuals facet, or the Ludi system [5] which generates game rules for two-player board games as part of the game design facet. Although such algorithms are beneficial for automatic generation of content, it is still rare that game characters are considered.

Web content is dispersed over the Internet in the form of blogs, microblogs, forums, wikis, social networks, review sites, and other Web applications which are currently disconnected from one another. The datasets created by these communities all contain information which can be used to generate or reuse game character content in games, but are not easily discoverable. The emerging Web of Data trend [3], where datasets are published in a standard form for easy interlinking, enables to essentially view the whole Web as one massive integrated database. Nevertheless, game character information is still not enriched with meta-structures that could be used both on the Web and also in games. With such rich meta-structures that add more meaning to game character content, this would enable Web content to be reused in games. Moreover, the representation of *semantically-enriched* and *semantically-interlinked* game character content would enable game character generators or game character asset managers to infer how characters can be interacted within the game world without having to rely on software development procedures that require laborious annotation of how each entity can be interacted within the game.

In this paper, we present our Game Character Ontology (GCO)[1] – a light-weight vocabulary for describing character information in video games. This ontology can be used to define character information extracted from various game character data sources such as Fandom[2] and Giant Bomb[3]. The remainder of this paper is as follows: Section 2 reviews current work on semantics in games and existing game ontologies. In Section 3 we present our motivations for developing a game character ontology. Section 4 presents a detailed explanation of our ontology including our approach taken to model this ontology, and it also provides an example of defining

[1] Game Character Ontology (GCO) – http://autosemanticgame.eu/ontologies/gco
[2] Fandom – http://fandom.wikia.com/games
[3] Giant Bomb – http://www.giantbomb.com/characters/

a game character using GCO. Section 5 concludes the paper by providing an overall discussion about the future steps of our work.

## 2 RELATED WORK

Semantics in games is still in its infancy and perhaps the closest attempt at using structured real-world data in games is the Data Adventures project [1, 2] which uses SPARQL queries on DBpedia to discover links between two or more individuals: the discovered links are transformed into adventure games, with entities of type "Person" becoming Non-Player Characters (NPCs) that the player can converse with, entities of type "City" becoming cities that the player can visit, and entities of type "Category" becoming books that the player can read. The advantages of using rich semantic information to automatically generate games are numerous [16] as more complex, open-world, non-linear, games incorporating very rich forms of interaction are possible (i.e. authentic sandbox games). Current work in using semantics in games focuses on the use of semantic information to generate game worlds or to describe interactions with game worlds such as the work in [8, 11, 17]. Although these provide useful insights in generic semantic models that describe interactions with game worlds, they do not offer vocabularies for describing game content such as game characters and they neither provide a generic approach for reusing Web content to generate games.

Attempts in game ontology creation are relevant to our approach, hence, we outline some key game-based ontologies currently existent. The Game Ontology Project [21] is a wiki-based knowledgebase that aims to provide elements of gameplay. However, this project does not take into consideration game characters. Moreover, it does not provide a vocabulary to be consumed by data described in RDF which could make it potentially useful for game character generation. The Digital Game Ontology [6] provides a general game ontology by aligning with the Music Ontology, and the Event and Timeline ontology, to provide concepts that describe digital games. However, the vocabulary is not available and in this regard, it is unclear what game concepts this vocabulary provides. The Ludo ontology [13] provides concepts that describe different aspects of serious games, however, it does not provide detailed concepts for describing game characters. The SALERO virtual character ontology [4] provides a generic ontology for describing characters in media production. Although it provides a generic model, it does not provide detailed concepts for defining game character information such as the species, race, character class, skills, weapons etc. The authors in [7] provide a generic ontology for defining RPG games and do not provide detailed concepts to describe game characters for other genre types. Finally, the Video Game Ontology (VGO) [12] provides concepts for defining interoperability amongst video games and the Game2Web ontology [14] focuses on linking game events and entities to social data. Although these vocabularies are useful for describing several aspects of game information, the ontologies are still limited to specific features and hence do not provide features for describing detailed game character information.

Our ontology was created since although similar classes and properties could be found in other ontologies, these do not imply that they are properties of fictional characters. For example most properties found in the FOAF vocabulary[4] are normally used for describing persons of type `foaf:Person`, and therefore, this vocabulary does not provide sufficient attributes to describe fictional game character information. Another example would be the classes and properties defined in DBpedia's ontology[5] such as `dbo:skinColor`, `dbo:hairColor` or `dbo:eyeColor` that describe physical features of real persons specified by the domain `dbo:Person` implying that the subject is a real person. Moreover, the Appearance Ontology[6] provides classes and properties for defining appearance and gender features of real persons specified by the domain `appearances:Person` implying that the subject is a real person. Therefore these classes and properties cannot be used to describe properties of fictional characters, and new classes and properties are required.

## 3 MOTIVATION

As mentioned in section 1, current social media platforms provide rich video game content which are not exploited for game content generation. Since games are complex and contain many types of content, in this paper we focus specifically on content describing game characters which is used in all the game facets. Game character content is available in wiki-based sites such as Fandom and Giant Bomb. As can be noted, such articles provide comprehensive detail about characters from which character models can be created. Other content can be found in user-based review sites such as reviews in Steam Discussions platform[7].

Suppose a framework that can automatically extract game character information from various social media platforms, and this content is inter-linked and semantically-annotated to provide comprehensive character models. These models would be described in RDF using the Game Character ontology (GCO) and would collectively create a game character dataset. This dataset would contain rich information about characters that can be reused in games and/or for generating new unconventional game characters from existing ones. This game character dataset would also provide character assets for generating characters using game designer applications. For example, if a designer requires a *Mage* in a particular game which s/he is designing, the designer could reuse the details of mage characters already existing in a game character dataset without having to re-create a mage from scratch. This would reduce the time and cost for generating video game characters in video games. Moreover, games could be designed to automatically generate (playable or non-playable) characters during gameplay from such game character dataset without the need of pre-creating characters in games. Furthermore, several games, such as the Baldur's Gate series[8], Neverwinter Nights series[9] and The Elders Scrolls series[10], provide users an in-game character generator to create their character (which they will play with) from a pre-determined set of character information manually created by developers. We envisage that in-game character generators will exploit a semantic-based game character dataset which would provide more character details

[4]FOAF – http://xmlns.com/foaf/0.1/
[5]DBpedia Ontology – http://dbpedia.org/ontology/
[6]Appearance Ontology – http://rdf.muninn-project.org/
[7]Steam Discussions – http://steamcommunity.com/discussions/
[8]Baldur's Gate Series – https://www.baldursgate.com
[9]Neverwinter Nights Series – http://www.bioware.com/en/games
[10]The Elder Scrolls Series – https://elderscrolls.bethesda.net/

from which users can choose from without requiring developers to manually create all this content. This will provide users to create unconventional characters during gameplay.

## 4 THE GAME CHARACTER ONTOLOGY (GCO)

The Game Character Ontology (GCO)[11] (illustrated in Fig. 1) provides a light-weight vocabulary for describing characters in video games. The process in modelling this ontology involved by first examining and extracting concepts from 100 different characters in highly-ranked *action-adventure* and *role-playing* video games described in fansites such as Fandom and Giant Bomb (amongst others). Then, these concepts were checked whether they could be described using existing ontologies, but as explained in section 2, most existing concepts define real persons rather than fictional game characters, and hence this ontology was created.

An example of a game character's information described using GCO can be found at http://autosemanticgame.eu/datasets/characters/Altair_ibn_La-Ahad. This example describes the information about the character named *Altaïr Ibn-La'Ahad* found in the *Assassin's Creed* series[12]. The information about this character was extracted from articles found in Fandom[13] and Giant Bomb[14].

Characters are defined using the main class `gco:Character` and could either be playable or non-playable characters, defined by the `gco:PlayableCharacter` or `gco:NonPlayableCharacter` subclasses respectively. Playable characters are characters controlled by the player whereas non-playable characters are controlled by AI. In some games, characters could be playable at some point in the game and non-playable at other points in the same game. In this regard, characters can be defined as instances of both the `gco:PlayableCharacter` and/or `gco:NonPlayableCharacter` subclasses. Moreover, characters could be of any type, from human-like characters to any type of creature, defined by the class `gco:Species` (see section 4.2). Furthermore, characters could be main protagonists or main antagonists in a game. Whether a character is a main protagonist or not is beyond the scope of this ontology since this is bound to a specific game rather than a character. Therefore, we model game characters irrespective of which game they are played or appeared in.

Although the ontology is designed based on *action-adventure* and *role-playing* video game characters such as characters in the *Assassin's Creed* series or *The Elder Scrolls* series, the ontology still can be used for defining characters in video games of other genre types (such as *first-person shooter* video games or *sports* video games) since the ontology provides detailed classes and properties that can be used for modelling any type of game character.

This section provides a comprehensive description of the classes and properties found in GCO.

### 4.1 Character Main Properties

The following are the main properties that describe different details and characteristics of a character.

- `gco:nickname` specifies a character's nickname.
- `gco:age` specifies a character's age value. This value indicates that the character does not age during a game, and the whole game is set during the year of the character's age.
- `gco:height` specifies a character's height value.
- `gco:weight` specifies a character's weight value.
- `gco:birthDate` specifies the character's birth date value as an `xsd:date`.
- `gco:deathDate` specifies a character's death date value as an `xsd:date`.
- `gco:birthYear` specifies a character's year of birth value.
- `gco:deathYear` specifies a character's year of death value.
- `gco:hasGender` specifies a gender of the character described using the `gco:Gender` class. This class defines a character's gender as either Male or Female by using the `gco:Male` or `gco:Female` subclasses respectively.
- `gco:skinColour`[15] specifies a skin colour tone defined by `dbo:Colour` which provides properties for describing the RGB colour model.
- `gco:hairColour` specifies a hair colour defined by `dbo:Colour`.
- `gco:facialHairColour` specifies a facial hair colour defined by `dbo:Colour`.
- `gco:eyeColour` specifies an eye colour defined by `dbo:Colour`.
- `gco:hasImage` specifies digital images that depict a character and is defined using `foaf:Image`.
- `gco:hasPersonality` describes various aspects of the character's personality defined by `gco:Personality`. The character's personality specifies a character's characteristic pattern of thinking, feeling and/or behaviour, for example whether the character is an angry (hot-tempered) and/or arrogant character.
- `gco:birthPlace` specifies a place of birth of a character defined by `gco:Place`.
- `gco:deathPlace` specifies a place of death of a character defined by `gco:Place`.
- `gco:livesIn` specifies the current place where a character lives defined by `gco:Place`.
- `gco:livedIn` specifies past place(s) where a character lived defined by `gco:Place`.
- `gco:currentOccupation` specifies the current professional occupation of a character defined by `gco:Occupation`.
- `gco:pastOccupation` specifies past professional occupation(s) held by a character defined by `gco:Occupation`.
- `gco:voicedBy` specifies the voice actor who represents the character's voice, and the attribute `xml:lang` could be used to define the language. The voice actor is defined by `foaf:Person`.
- `gco:appearsIn` specifies the game(s) in which the character appears defined by `cgo:Game` defined in the Core Game Ontology (CGO)[16] – an ontology for defining core concepts of video games.
- `gco:hasSpecies` specifies the species of a character defined using `gco:Species` explained in subsection 4.2.

[11] Game Character Ontology (GCO) – http://autosemanticgame.eu/ontologies/gco
[12] Assassin's Creed Series – http://assassinscreed.ubi.com
[13] *Altaïr* on Fandom – http://assassinscreed.wikia.com/wiki/Altaïr_Ibn-La'Ahad
[14] *Altaïr* on Giant Bomb – https://www.giantbomb.com/altair-ibn-laahad/3005-48/

[15] Note: the attributes `gco:skinColour`, `gco:hairColour`, `gco:facialHairColour` and `gco:eyeColour` fall under the `gco:Appearance` class.
[16] Core Game Ontology (CGO) – http://autosemanticgame.eu/ontologies/cgo

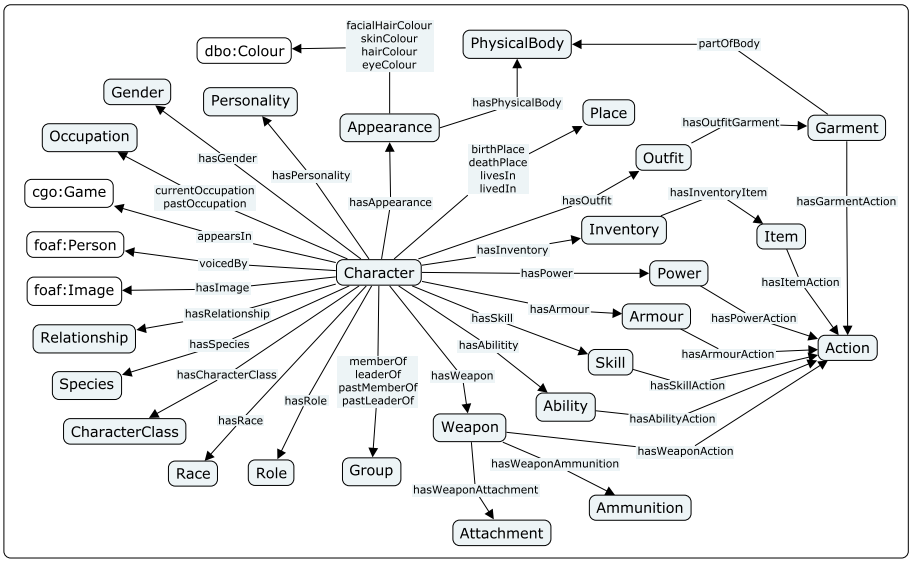


**Figure 1: Overview of the Game Character Ontology (GCO)**

- `gco:hasRace` specifies a more specific classification of the character's species defined using `gco:Race` explained in subsection 4.2.
- `gco:hasCharacterClass` specifies another type of classification for characters that represent archetypes or careers defined by `gco:CharacterClass` explained in subsection 4.2.
- `gco:hasRole` specifies a role of a character defined by `gco:Role` explained in subsection 4.3.
- `gco:hasSkill` specifies a skill of a character defined by `gco:Skill` explained in subsection 4.4.
- `gco:hasAbility` specifies an ability of a character defined by `gco:Ability` explained in subsection 4.4.
- `gco:hasPower` specifies a power of a character defined using `gco:Power` explained in subsection 4.4.
- `gco:hasWeapon` specifies a weapon that a character can use which is defined by `gco:Weapon` explained in subsection 4.5.
- `gco:hasArmour` specifies a piece of armour that a character can wear or use to protect himself/herself from any damage, which is defined by `gco:Armour` explained in subsection 4.5.
- `gco:hasInventory` specifies an item or set of items that a character can use which is defined using `gco:Inventory` explained in subsection 4.5.
- `gco:hasOutfit` specifies an outfit (consisting of a set of garments) a character can wear which is defined by the class `gco:Outfit` explained in subsection 4.5.
- `gco:hasRelationship` specifies a relationship amongst characters defined by `gco:Relationship` explained in subsection 4.6.
- `gco:memberOf` specifies that a character is a current member of a particular group defined using `gco:Group` explained in section 4.7.
- `gco:leaderOf` specifies that a character is a current leader of a particular group defined by `gco:Group` explained in section 4.7.
- `gco:pastMemberOf` specifies that a character is a past member of a particular group defined by `gco:Group` explained in section 4.7.
- `gco:pastLeaderOf` specifies that a character is a past leader of a particular group defined by `gco:Group` explained in section 4.7.
- `gco:onlineArticle` specifies a reference to a URL of an on-line article describing a character, for example an article from fansites such as Fandom.

The following are the main classes of GCO and they all contain the following properties: (1) `dcterms:title` that defines a name for the resource, and (2) `dcterms:description` that defines a description for the resource.

## 4.2 Species, Race and CharacterClass

*4.2.1 Species.* defines fictional representations of generic character classifications for example the *Human* species or even fictional species such as *Orc* or *Elf*, and it contains the following properties:

- `gco:hasSpeciesSkill` defines a default skill (defined by `gco:Skill`) that a character classified by a particular species can perform.
- `gco:hasSpeciesAbility` defines a default ability (defined by `gco:Ability`) that a character classified by a particular species can perform.
- `gco:hasSpeciesPower` defines a default power (defined by `gco:Power`) that a character classified by a particular species can perform.

*4.2.2 Race.* defines more specific character classifications than the species classification, for example the *Caucasian* race or the *High Elf* race, and it contains the following properties:

- gco:raceOf specifies that a particular race falls under a particular species (defined by gco:Species). For example the *High Elf* race falls under the *Elf* species. All the properties of the defined species linked to a race are automatically propagated to all the characters classified under that particular race.
- gco:hasRaceSkill defines a default skill (defined by the class gco:Skill) that a character classified by a particular race can perform.
- gco:hasRaceAbility defines a default ability (defined by gco:Ability) that a character classified by a particular race can perform.
- gco:hasRacePower defines a default power (defined by gco:Power) that a character classified by a particular race can perform.

*4.2.3 CharacterClass.* defines a more specific classification of characters based on archetypes and/or careers, for example the character class *Assassin*. Character classes are independent from race and/or species which means that any race and/or species could be of any character class. In other words, it is possible to have two characters of the same character class but are of different race and /or species. For example a character of race *Caucasian* and another character of race *High Elf* can both be of character class *Assassin*. Moreover, a character could be defined with more than one class, for example a character can be assigned to a *Mage* class and *Fighter* class. Character classes contain the following properties:

- gco:hasCharacterClassSkill defines a default skill (defined by gco:Skill) that a character classified by a particular character class can perform.
- gco:hasCharacterClassAbility defines a default ability (defined by gco:Ability) that a character classified by a particular character class can perform.
- gco:hasCharacterClassPower defines a default power (defined by gco:Power) that a character classified by a particular character class can perform.

All the properties of skills, abilities and/or powers assigned to species, races or character classes are propagated to all the characters classified under the respective species, race or character class.

*4.2.4 Definition 1: Propagation of Skills, Abilities and Powers to Characters.* The following defines the propagation of skills, abilities and powers from species, races or character classes to characters.

Let *Sp* be a species, *Ra* a race, *Cc* a character class, *Ch* a character, *Sk* a skill, *Ab* an ability and *Po* a power. Let $AssignSkill(Sp, Sk)$ mean that *Sp* is assigned *Sk*, $AssignSkill(Ra, Sk)$ mean that *Ra* is assigned *Sk*, $AssignSkill(Cc, Sk)$ mean that *Cc* is assigned *Sk*, $AssignAbility(Sp, Ab)$ mean that *Sp* is assigned *Ab*, $AssignAbility(Ra, Ab)$ mean that *Ra* is assigned *Ab*, $AssignAbility(Cc, Ab)$ mean that *Cc* is assigned *Ab*, $AssignPower(Sp, Po)$ mean that *Sp* is assigned *Po*, $AssignPower(Ra, Po)$ mean that *Ra* is assigned *Po*, $AssignPower(Cc, Po)$ mean that *Cc* is assigned *Po*, $AssignSpecies(Ch, Sp)$ mean that *Ch* is assigned *Sp*, $AssignSpecies(Ra, Sp)$ mean that *Ra* is assigned *Sp*, $AssignRace(Ch, Ra)$ mean that *Ch* is assigned *Ra*, $AssignClass(Ch, Cc)$ mean that *Ch* is assigned *Cc*, $AssignSkill(Ch, Sk)$ mean that *Ch* is assigned *Sk*, $AssignAbility(Ch, Ab)$ mean that *Ch* is assigned *Ab* and $AssignPower(Ch, Po)$ mean that *Ch* is assigned *Po*.

Propagation of skills is defined as follows:

$$\begin{aligned}\forall Sk((&AssignSkill(Sp,Sk) \wedge AssignSpecies(Ch,Sp)) \vee \\ &(AssignSkill(Ra,Sk) \wedge AssignRace(Ch,Ra)) \vee \\ &(AssignSkill(Sp,Sk) \wedge AssignSpecies(Ra,Sp) \wedge AssignRace(Ch,Ra)) \vee \\ &(AssignSkill(Cc,Sk) \wedge AssignClass(Ch,Cc)) \Rightarrow AssignSkill(Ch,Sk))\end{aligned} \quad (1)$$

Propagation of abilities is defined as follows:

$$\begin{aligned}\forall Ab((&AssignAbility(Sp,Ab) \wedge AssignSpecies(Ch,Sp)) \vee \\ &(AssignAbility(Ra,Ab) \wedge AssignRace(Ch,Ra)) \vee \\ &(AssignAbility(Sp,Ab) \wedge AssignSpecies(Ra,Sp) \wedge AssignRace(Ch,Ra)) \vee \\ &(AssignAbility(Cc,Ab) \wedge AssignClass(Ch,Cc)) \Rightarrow AssignAbility(Ch,Ab))\end{aligned} \quad (2)$$

Propagation of powers is defined as follows:

$$\begin{aligned}\forall Po((&AssignPower(Sp,Po) \wedge AssignSpecies(Ch,Sp)) \vee \\ &(AssignPower(Ra,Po) \wedge AssignRace(Ch,Ra)) \vee \\ &(AssignPower(Sp,Po) \wedge AssignSpecies(Ra,Sp) \wedge \\ &AssignRace(Ch,Ra)) \vee (AssignPower(Cc,Po) \\ &\wedge AssignClass(Ch,Cc)) \Rightarrow AssignPower(Ch,Po))\end{aligned} \quad (3)$$

## 4.3 Role

The Role class defines specific representations of roles that characters can change into. For example, although a character could be of character class *Assassin*, the character could switch role to a *Slave* role having different outfits, skills, abilities, weapons, armour etc. without changing the main occupation or class of the character – i.e. *Assassin*. This class contains the following properties:

- gco:hasRoleSkill defines that a character can only use a particular skill (defined by gco:Skill).
- gco:hasRoleAbility defines that a character can only use a particular ability (defined by gco:Ability).
- gco:hasRolePower defines that a character can only use a particular power (defined by gco:Power).
- gco:hasRoleWeapon defines that a character can only use a particular weapon (defined by gco:Weapon).
- gco:hasRoleArmour defines that a character can only use a particular armour (defined by gco:Armour).
- gco:hasRoleInventory defines that a character can only use a particular inventory (defined by gco:Inventory).
- gco:hasRoleOutfit defines an outfit (defined by gco:Outfit) assigned to a role.
- gco:hasRoleInGroup defines that a particular role is used in a group – i.e. a character's role in a group.

All the properties of the defined skills, abilities and powers are automatically propagated to all the characters assigned to a particular role.

*4.3.1 Definition 2: Propagation of Skills, Abilities and Powers from Roles to Characters.* Once the skills, abilities and/or powers are propagated from the species, races and/or character classes to characters, these are then checked to ensure whether they are assigned to the character role. If a character does not contain any roles, then the assigned skills, abilities and/or powers are held. Otherwise, a character will only be assigned those skills, abilities and/or powers assigned to a particular role, as defined below.

Let *Ro* be a role, *Ch* a character, *Sk* a skill, *Ab* an ability and *Po* a power. Let *AssignSkill*(*Ro*, *Sk*) mean that *Ro* is assigned *Sk*, *AssignAbility*(*Ro*, *Ab*) mean that *Ro* is assigned *Ab*, *AssignPower*(*Ro*, *Po*) mean that *Ro* is assigned *Po*, *AssignRole*(*Ch*, *Ro*) mean that *Ch* is assigned *Ro*, *AssignSkill*(*Ch*, *Sk*) mean that *Ch* is assigned *Sk*, *AssignAbility*(*Ch*, *Ab*) mean that *Ch* is assigned *Ab* and *AssignPower* (*Ch*, *Po*) mean that *Ch* is assigned *Po*.

Propagation of skills from role to character is defined as follows:

$$\forall Sk(AssignSkill(Ro,Sk) \land AssignRole(Ch,Ro) \Rightarrow AssignSkill(Ch,Sk)) \quad (4)$$

Propagation of abilities from role to character is defined as follows:

$$\forall Ab(AssignAbility(Ro,Ab) \land AssignRole(Ch,Ro) \Rightarrow AssignAbility(Ch,Ab)) \quad (5)$$

Propagation of powers from role to character is defined as follows:

$$\forall Po(AssignPower(Ro,Po) \land AssignRole(Ch,Ro) \Rightarrow AssignPower(Ch,Po)) \quad (6)$$

## 4.4 Skill, Ability and Power

*4.4.1 Skill.* defines active actions (activated by a player) which characters can perform to complete a specific task in a game, for example running, climbing and grabbing, and it contains the following properties:

- `gco:defaultSkillHitPoints` defines default initial skill hit points.
- `gco:hasSkillAction` defines a default action bound to a skill that when executed produces a result.

*4.4.2 Ability.* defines passive actions that are applied automatically and are not activated by a player, for example "increase the character's armour by 20%" or "enhance sniper rifle hit points once the ability is obtained". Abilities could have restrictions on their usage for example "used only once per day". This class contains the following properties:

- `gco:defaultAbilityHitPoints` defines default initial ability hit points.
- `gco:hasAbilityRestriction` defines a restriction on using an ability.
- `gco:hasAbilityAction` defines a default action bound to an ability that when executed produces a result.

*4.4.3 Power.* defines unique actions that are activated by a player which normally consist of super-natural actions in nature for example magic spells or sources of energy that power up armour. This class contains the following properties:

- `gco:defaultPowerHitPoints` defines default initial power hit points.
- `gco:hasPowerAction` defines a default action bound to a power that when executed produces a result.

All actions described above are defined by the `gco:Action` class which defines the actions as RIF [9] rules. These rules can be used by a game engine to execute such actions. Moreover, the skills, abilities and/or powers could increase in hit points and could level up a character during a game's specific gameplay. Therefore, such increase points properties and/or character level up properties are beyond the scope of this ontology. Furthermore, the current skill, ability and/or power points and the current level of a character are bound to specific players and gameplay. Therefore, these properties are also beyond the scope of this ontology.

## 4.5 Weapon, Armour, Inventory and Outfit

*4.5.1 Weapon.* defines a weapon which characters can use or carry and it contains the following properties:

- `gco:defaultWeaponHitPoints` defines default weapon hit points.
- `gco:singleHandedWeapon` defines a boolean to specify whether the weapon is single-handed or not. A weapon cannot be both single-handed or double-handed but a character could have two single-handed weapons in separate hands.
- `gco:doubleHandedWeapon` defines a boolean to specify whether the weapon is double-handed or not. A weapon cannot be both single-handed and double-handed, and a character can only use one double-handed weapon at a time (if the character has two hands). This normally requires that a character has a double-handed skill which enables the character to yield a double-handed weapon, but this restriction should be defined in the game play rules of a game and therefore this is beyond the scope of this ontology.
- `gco:hasWeaponAction` defines a default action bound to a weapon.

*4.5.2 Armour.* defines a piece of armour which characters can use or wear to deflect any damage (i.e. minimise hit points) inflicted by opponents, and it contains the following properties:

- `gco:armourWearable` defines a boolean whether an armour can be wearable or not.
- `gco:defaultArmourDeflectPoints` defines the default points an armour can deflect.
- `gco:defaultArmourHitPoints` defines the default hit points an armour can cause.
- `gco:hasArmourAction` defines a default action bound to an armour.

*4.5.3 Inventory.* contains the `gco:hasInventoryItem` property that defines an item such as health potions which characters can use or carry. Items are defined by the `gco:Item` class that contains the following properties:

- `gco:defaultItemPoints` defines default points an item can cause.
- `gco:hasItemAction` defines a default action bound to an item.

*4.5.4 Outfit.* defines a piece of garment or a set of garments characters can wear and it contains the gco:hasOutfitGarment property which specifies a piece of garment defined by the gco:Garment class. The gco:Garment class contains the following properties:

- gco:partOfBody defines the part of the body where the garment is worn.
- gco:overGarmentOf defines that a piece garment is over another piece of garment.
- gco:underGarmentOf defines that a piece of garment is under another piece of garment.
- gco:hasGarmentColour defines the colour of the garment (defined by dbo:Colour).
- gco:defaultGarmentHitPoints defines default hit points a garment can cause (if any).
- gco:defaultGarmentDeflectPoints defines default points a garment can deflect (if any).
- gco:hasGarmentAction defines a default action bound to a garment (if any).

All actions are defined by the gco:Action class which defines the actions as RIF [9] rules. Moreover, weapons, armour, inventory items and outfits could be upgraded which could increase a character's level. Furthermore, games and their respective gameplay could have restrictions on the number of weapons, armour, inventory items and outfits which a character could use and/or carry. Since these properties are bound to a specific game and gameplay they are beyond the scope of this ontology.

## 4.6 Relationship

The Relationship class defines various types of relationships which characters have with other characters. Although some of the relationships are similar to the properties defined in the Relationship Ontology[17], this ontology however defines relationships amongst real persons (defined as foaf:Person) and hence cannot be used to define relationships amongst fictional characters of type gco:Character. Therefore, gco:Relationship class provides the following relationship properties:

- gco:acquaitanceOf defines that a character has met and knows slightly another character but they lack friendship.
- gco:ancestorOf defines that a character is an ancestor of another character.
- gco:apprenticeTo defines that a character serves and learns from another character.
- gco:closeFriendOf defines that two characters share a close friendship.
- gco:collaboratesWith defines that two characters collaborate with each other to accomplish a common goal.
- gco:colleagueOf defines that a character works with another character within a group but it does not mean that they work together to achieve a common goal.
- gco:descendantOf defines that a character is a descendant of another character.
- gco:employedBy defines that a character is employed by another character.
- gco:employerOf defines that a character is an employer of another character.
- gco:enemyOf defines that two characters are enemies of each other.
- gco:engagedTo defines that two characters are engaged with each other.
- gco:friendOf defines that a character is a friend of another character.
- gco:influencedNegativelyBy defines that a character is negatively influenced by another character.
- gco:influencedPositivelyBy defines that a character is positively influenced by another character.
- gco:isChild defines that a character is a child of another character, defined by the gco:Child class. This class contains the following properties: (1) gco:sonOf defines that a character is the son of another character, (2) gco:daughterOf defines that a character is the daughter of another character.
- gco:isGrandChild defines that a character is a grand child of another character, defined by the gco:GrandChild class. This class contains the following two properties: (1) gco:grandSonOf defines that a character is the grand son of another character, (2) gco:grandDaughterOf defines that a character is the grand daughter of another character.
- gco:isGrandParent defines that a character is a grand parent of another character, defined by the gco:GrandParent class. This class contains the following two properties: (1) gco:grandFatherOf defines that a character is the grand father of another character, (2) gco:grandMotherOf defines that a character is the grand mother of another character.
- gco:isParent defines that a character is a parent of another character, defined by the gco:Parent class. This class contains the following two properties: (1) gco:fatherOf defines that a character is the father of another character, (2) gco:motherOf defines that a character is the mother of another character.
- gco:isSibling defines that a character is a sibling of another character, defined by the gco:Sibling class. This class contains the following two properties: (1) gco:brotherOf defines that a character is the brother of another character, (2) gco:sisterOf defines that a character is the sister of another character.
- gco:isSpouse defines that a character is a spouse of another character, defined by the gco:Spouse class. This class contains the following two properties: (1) gco:husbandOf defines that a character is the husband of another character, (2) gco:wifeOf defines that a character is the wife of another character.
- gco:knowsOf defines that a character knows of another character but does not necessarily mean that they have met or have any other type of friendship.
- gco:livesWith defines that a character lives with another character but does not necessarily mean that they are married, or are partners or have any other type of friendship.

[17] Relationship Vocabulary – http://vocab.org/relationship/

- gco:mentorOf defines that a character is a mentor of another character.
- gco:neighbourOf defines that a character lives near another character.
- gco:partnerOf defines that two characters share an intimate relationship but who are not married to each other.
- gco:petOf defines that a character is a pet of another character.
- gco:slaveOf defines that a character is a slave of another character.
- gco:slaveOwnerOf defines that a character is the owner of another character who is a slave.

### 4.7 Group

The Group class defines various types of groups in which characters could be a part of – from formal groups such as the *Brotherhood of Assassins* or the *Templar Order* in the *Assassin's Creed* series to informal groups were different characters team up to solve a common goal such as in the *Uncharted*[18] or *Tomb Raider*[19] series. This class provides the following properties:

- gco:hasGroupLogo defines a symbolic image (defined by foaf:Image) that symbolises a group which could be a coat of arms, a seal, a flag etc.
- gco:hasMember specifies who is a current member in a group.
- gco:hasPastMember specifies past members (and are not current members) in a group.
- gco:hasLeader specifies who is a current leader of a group.
- gco:hasPastLeader specifies past leaders of a group.

Moreover, the gco:Group class defines 67 different sub-classes which represent different types of groups, for example gco:Brotherhood or gco:Order groups.

## 5 CONCLUSION

In this paper we presented our game character ontology for generating game characters via semantic information extracted from diverse Web content. In contrast to current automated game generation processes such as traditional procedural content generation (PCG) practices, our approach enables the use of massive amounts and dissimilar types of content from online sources. This allows content to be automatically generated whilst taking into consideration player models derived from user information stored across various online datasets [19] thereby realising a semantically-enriched version of the experience-driven PCG framework [20]. The ontology presented in this paper can be used to create game characters to be used in games, which are expected to appeal to the entire community or to specific parts of the community, based for instance on demographics or skill or interests collected from user's steam achievements or favoured games, respectively. On the other hand, an indirect model of player engagement with specific types of content can be gleaned from the mostly user-generated Fandom pages. Pages with popular characters and locations or challenging game levels are expected to have more textual contributions (due to being updated more often by more people). This can be used to create content similar to existing game content popular in one or more Fandom user communities.

With the novel approach proposed in this paper we envisage not only the generation of characters in digital games autonomously but also the creation of games that are perceived as being unconventional and unexpected, yet engaging and playable.

[18]Uncharted series – http://www.unchartedthegame.com
[19]Tomb Raider series – https://www.tombraider.com

## ACKNOWLEDGMENTS

The research work disclosed in this publication is partially funded by the REACH HIGH Scholars Programme — Post-Doctoral Grants. The grant is part-financed by the European Union, Operational Programme II — Cohesion Policy 2014-2020 Investing in human capital to create more opportunities and promote the wellbeing of society — European Social Fund.

## REFERENCES

[1] G. A. Barros, A. Liapis, and J. Togelius. Playing with Data: Procedural Generation of Adventures from Open Data. In *International Joint Conference of DiGRA and FDG*, DiGRA-FDG'16, 2016.
[2] G. A. Barros, A. Liapis, and J. Togelius. Who Killed Justin Bieber? Murder Mystery Generation from Open Data. In *International Conference on Computational Creativity*, ICCC'16, 2016.
[3] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked Data on the Web (LDOW2008). In *17th International Conference on World Wide Web*, WWW'08, 2008.
[4] T. BÃijrger, P. Hofmair, and G. Kienast. The salero virtual character ontology. In *Proceedings of the First Workshop on Semantic 3D Media*, 2008.
[5] C. Browne and F. Maire. Evolutionary Game Design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.
[6] J. T. C. Chan and W. Y. F. Yuen. Digital Game Ontology: Semantic Web Approach on Enhancing Game Studies. In *International Conference on Computer-Aided Industrial Design and Conceptual Design*, CAID/CD 2008, 2008.
[7] B. O. Durić and M. Konecki. Specific owl-based rpg ontology. In *Central European Conference on Information and Intelligent Systems*, 2015.
[8] J. Kessing, T. Tutenel, and R. Bidarra. Designing semantic game worlds. In *Workshop on Procedural Content Generation in Games*, PCG'12. ACM, 2012.
[9] H. B. Leora Morgenstern, Chris Welty and G. Hallmark. Rule Interchange Format (RIF) Primer. https://www.w3.org/TR/rif-primer/.
[10] A. Liapis, G. N. Yannakakis, and J. Togelius. Computational game creativity. In *Fifth International Conference on Computational Creativity*, ICCC'14, 2014.
[11] R. Lopes and R. Bidarra. A semantic generation framework for enabling adaptive game worlds. In *International Conference on Advances in Computer Entertainment Technology*. ACM, 2011.
[12] J. Parkkila, F. Radulovic, D. Garijo, M. Poveda-Villalón, J. Ikonen, J. Porras, and A. Gómez-Pérez. An ontology for videogame interoperability. *Multimedia Tools and Applications*, pages 1–20, 2016.
[13] O. R. Rocha and C. Faron-Zucker. Ludo: An Ontology to Create Linked Data Driven Serious Games. In *ISWC 2015 - Workshop on LINKed EDucation*, LINKED'15, 2015.
[14] O. Sacco, M. Dabrowski, and J. G. Breslin. Linking in-game events and entities to social data on the web. In *Games Innovation Conference (IGIC), 2012 IEEE International*, pages 1–4, Sept 2012.
[15] O. Sacco, A. Liapis, and G. N. Yannakakis. A Holistic Approach for Semantic-Based Game Generation. In *Proceedings of the IEEE Computational Intelligence and Games Conference*, CIG16, 2016.
[16] T. Tutenel, R. Bidarra, R. M. Smelik, and K. J. D. Kraker. The Role of Semantics in Games and Simulations. *Computers in Entertainment*, 6(4):57:1–57:35, Dec. 2008.
[17] T. Tutenel, R. M. Smelik, R. Bidarra, and K. J. de Kraker. Using Semantics to Improve the Design of Game Worlds. In *Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'09, 2009.
[18] I. D. Visualization. Speedtree., 2010.
[19] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. Andre. Player Modeling. In *S. M. Lucas, et al. eds. Artificial and Computational Intelligence in Games*. s.l.: Dagstuhl Seminar, 2013.
[20] G. N. Yannakakis and J. Togelius. Experience-driven procedural content generation (extended abstract). In *International Conference on Affective Computing and Intelligent Interaction*, ACII'15, 2015.
[21] J. P. Zagal and A. Bruckman. The Game Ontology Project: Supporting Learning While Contributing Authentically to Game Studies. In *8th International Conference on International Conference for the Learning Sciences*, ICLS'08, 2008.