Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

Отчёт по лабораторной работе №2
по дисциплине
«Проектирование баз знаний»

Выполнил студент гр. 121702
Д.В.Промчук

Проверила:
Н.Г. Липницкая

Минск 2024

Тема: Разработка и выравнивание онтологий.

Цель: Приобрести навыки разработки онтологий предметных областей.

Задачи:

1. Изучить принципы создания онтологий (см. Материалы для ознакомления)
2. Выбрать инструментальное средство для разработки онтологии (например, Protege).
3. Тему онтологии использовать из Лабораторной работы №1 (для интеллектуальной карты). Найти 2-3 онтологии совпадающих или близких к предметной области.
4. Разработать онтологию и загрузить существующие.
    1. Разработать онтологию по выбранной предметной области используя инструментальное средство (например, Protege). Предусмотреть описание не менее 10 классов сущностей выбранной предметной области, у каждого класса не менее 2 слотов, у каждого класса не менее 2 экземпляров
    2. Готовые онтологии (пункт 3) загрузить в одну онтологию.
5. Сделать запросы к онтологии
    1. Создать к разработанной онтологии 5 различных запросов.
    2. Сделать 3 запроса, показывающих использование информации из различных онтологий (SparQL).
6. По результатам работы оформить отчет: описать все выделенные классы, слоты онтологии, описать запросы и ответы, а также кратко описать функционал выбранного инструментального средства для разработки онтологии.

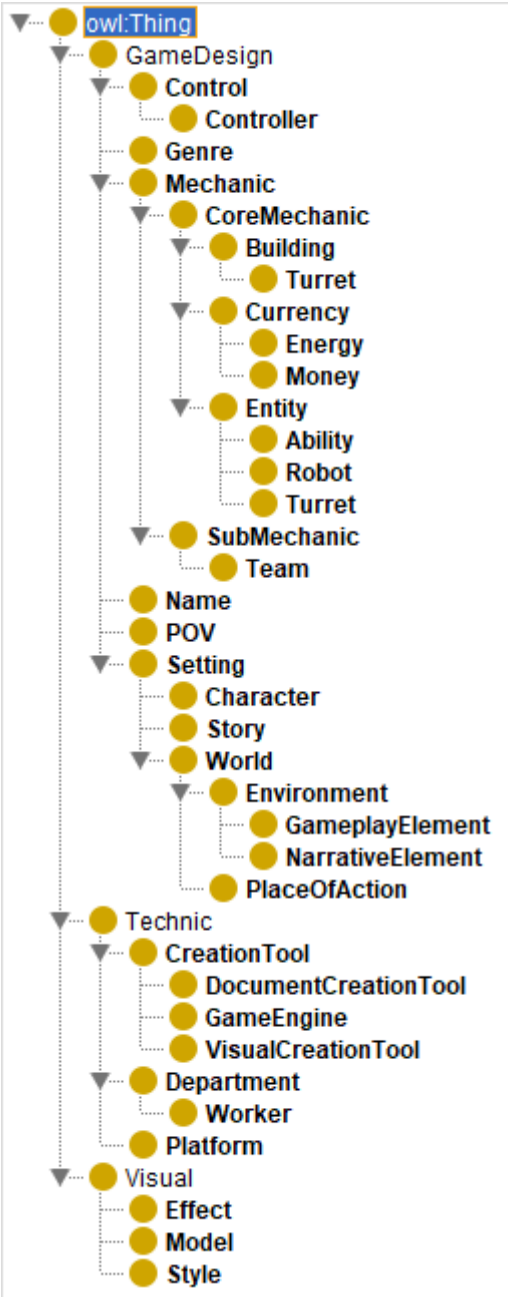
Вариант (предметная область): Онтология понятий и отношений компьютерной игры Sustain the Strain.

Выполнение заданий:

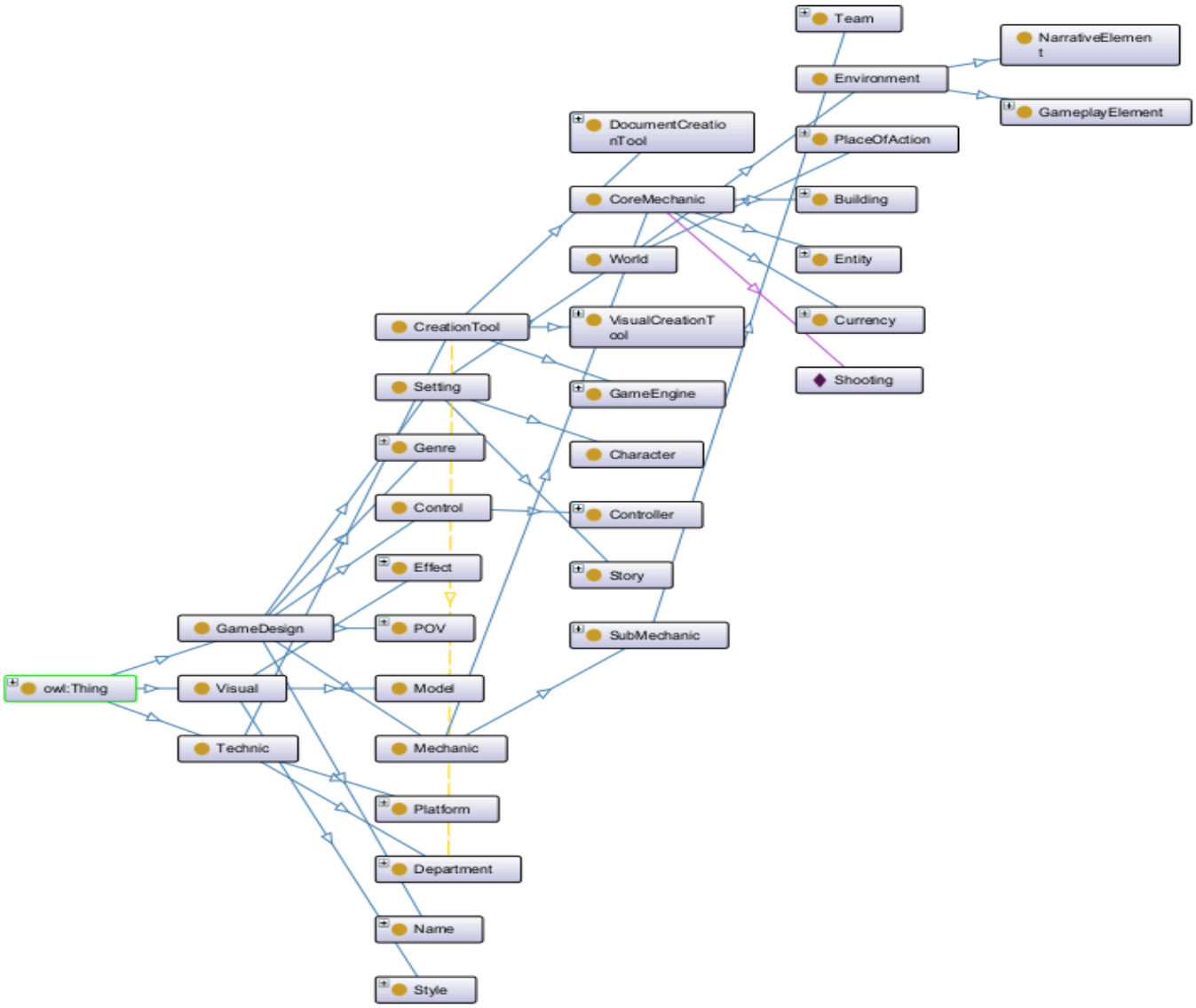
Для выполнения данной лабораторной работы использовались следующие материалы:
https://ceur-ws.org/Vol-3579/paper15.pdf
https://oa.upm.es/72411/1/REV_JCR_42_C.pdf
https://www.youtube.com/watch?v=DZCCfr4D8sk
https://www.youtube.com/watch?v=FvGndkpa
https://github.com/Stefano-Angelo-Rizzo/VideOWL
https://www.gameontology.com/index.php/Main_Page
https://www.pesc.coppe.ufrj.br/uploadfile/publicacao/2565.pdf
http://purl.org/net/VideoGameOntology
https://archivo.dbpedia.org/info?o=https://collection.rcgs.jp/terms

Для создания онтологии по предметной области был выбран инструментарий по созданию онтологий под названием Protege. Protege является мощным инструментом создания онтологий, поддерживающим популярные форматы онтологий, такие как OWL и RDF. Большим плюсом является возможность работать с данным инструментом онлайн, без необходимости его локальной установки. Помимо этого, данный инструмент был выбран по причине его широкой поддержки и наиболее частым обновлениям по сравнению с его аналогами (для данного инструмента последний выпуск датируется октябрём 2023 года, в то время как у остальных подобных инструментов обновления могут не выходить годами).

Структура классов реализованной онтологии представляет собой следующее:



Protege позволяет наглядно отображать онтологию в виде графа во вкладке OntoGraf.



Класс Thing является классом по умолчанию и не может быть изменён или удалён. Этот класс - основа всех следующих реализуемых классов.
Подклассами класса Thing являются классы GameDesign (игровой дизайн, те логические правила по которым существует игра), Technic (техническая сторона проекта и его реализации), Visual (визуальная составляющая проекта).
Далее в иерархическом порядке будут описаны классы онтологии.

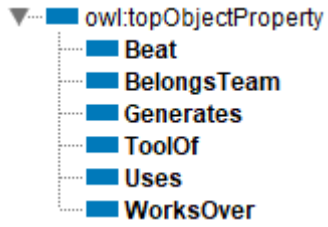
GameDesign
- Control - управление играет важную часть восприятия игры
  - Controller - то устройство, которым будет управлять игрок
- Genre - жанр игры
- Mechanic - механика, логика взаимодействия игровых объектов
  - CoreMechanic - механика, являющаяся основной для игры
    - Building - механика строений
      - Turret - турель
    - Currency - механика различной внутриигровой валюты
      - Energy - энергия(электричество)
      - Money - деньги(в нашем случае металл)
    - Entity - игровая сущность, основная логическ. единица взаим-я
      - Ability - особая способность, применяемая игроком
      - Robot - робот
      - Turret - турель
  - SubMechanic - дополнительная механика, раскрывающая основные
    - Team - механика противодействия команд
- Name - название проекта
- POV - “точка зрения игрока”, позиция с которой игрок видит действия игры
- Setting - среда, в которой происходит основное действие игры
  - Character - основное взаимодействующее лицо в игре
  - Story - история мира игры
  - World - мир, в котором проходит действие игры
    - Environment - окружение, локация основных действий
      - GameplayElement - окружение, с которым взаимодействует игрок
      - NarrativeElement - окружение, с которым игрок не взаимодействует
    - PlaceOfAction - территория основных действий игры

Technic
- CreationTool - инструмент создания какой-либо составляющей игры
  - DocumentCreationTool - инструмент создания документации для пр-ва проекта
  - GameEngine - инструмент для создания и воспроизведения самой игры
  - VisualCreationTool - инструмент для создания визуальных элементов
- Department - отдел разработки
  - Worker - работник отдела разработки
- Platform - целевое устройство, на котором будет воспроизводиться игра

Visual
- Effect - визуальный эффект (взрывы и т.д.)
- Model - модель игрового персонажа
- Style - визуальный стиль

В онтологиях свойства объектов делятся на два типа: объектные (Object Properties) и информационные (Data Properties). Доменами объектных свойств являются объекты различных классов. У информационных свойств первым доменом является объект, а вторым - какой-либо базовый тип (например, число).

Реализованная онтология включает в себя следующие объектные свойства:



owl:topObjectProperty - базовое объектное свойство

Beat [Entity-Entity] – указывает, что одна игровая сущность сражается с другой

BelongsTeam [Entity-Team] – указывает, какой команде принадлежит игровая сущность

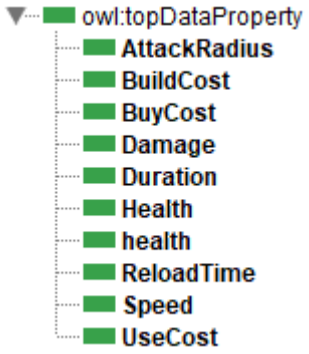
Generates [Building-Currency] – указывает, какую валюту производит постройка

ToolOf [CreationTool-Department] – указывает, какой отдел использует данный инструмент разработки

Uses [Entity-Currency] – указывает, какую валюту использует игровая сущность для своей работы

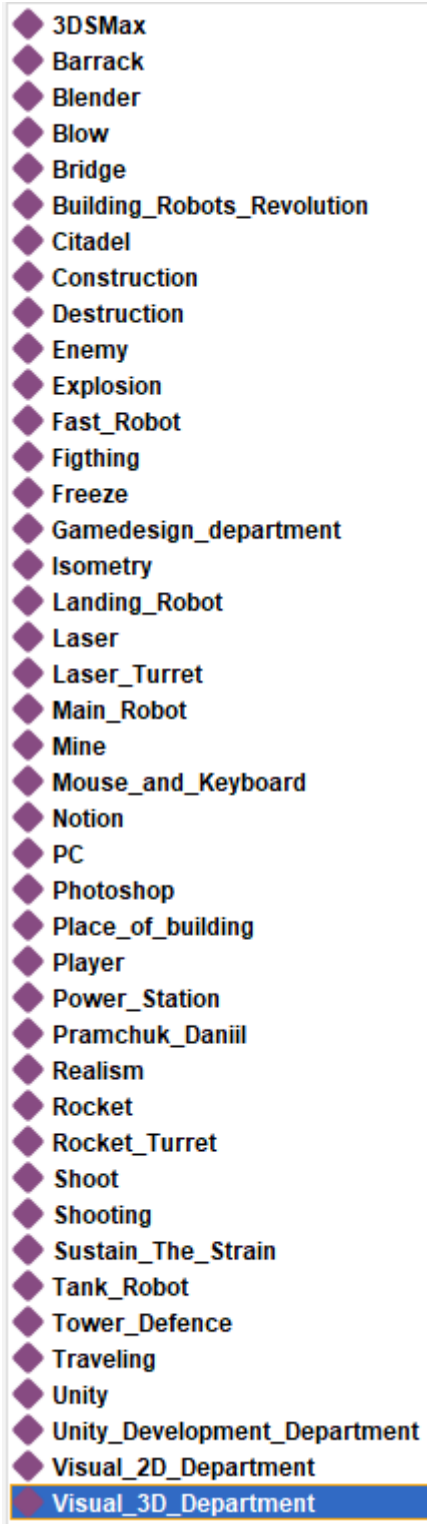
WorksOver [Worker-owl:Thing] – указывает, над какой составляющей проекта работает работник

Реализованная онтология включает в себя следующие информационные свойства:



owl:topDataProperty – базовое информационное свойство
AttackRadius [Entity-integer] – радиус атаки игровой сущности
BuildCost [Building-integer] – стоимость постройки здания
BuyCost [CreationTool-integer] – стоимость покупки инструмента разработки
Damage [Entity-integer] – урон игровой сущности
Duration [Effect-float] – длительность действия эффекта
Health [Robot-integer] – количество единиц жизни робота
health [Building-integer] – количество единиц жизни постройки
ReloadTime [Ability-float] – время перезарядки способности
Speed [Robot-integer] – скорость перемещения робота
UseCost [Ability-integer] – стоимость применения способности

Список объектов реализованной онтологии, представляет собой список, содержащий какие-либо свойства проекта(жанр, платформа и др в единичном экземпляре, а так же представителей различных классов онтологии)



Запросы к онтологии:
Для начала следует отметить что для сокращения количества текста без потери информативности, во всех запросах использовался следующий набор импортов:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX onto: <https://www.semanticweb.org/dansh/ontologies/2024/3/SustainTheStrain#>
```

1. Выбрать все сочетания субъект-объект онтологии, связанные каким либо свойством:

   Запрос:

   ```
   SELECT ?Subject ?Object WHERE {
   ?Subject rdf:type ?Object. }
   ```

   Результат (первые несколько позиций):

?Subject	?Object
onto:Barrack	onto:Setting
onto:Barrack	onto:CoreMechanic
onto:Barrack	onto:Entity
onto:Barrack	owl:Thing
onto:Blender	onto:CreationTool
onto:Blender	onto:VisualCreationTool
onto:Blender	onto:Technic
onto:Blender	owl:Thing
onto:Blow	onto:Visual
onto:Blow	onto:Effect
onto:Blow	owl:Thing
onto:Bridge	onto:GameplayElement
onto:Bridge	onto:GameDesign
onto:Bridge	onto:World
onto:Bridge	onto:Environment
onto:Bridge	onto:Setting
onto:Bridge	owl:Thing
onto:Building_Robots_Revolut...	onto:GameDesign
onto:Building_Robots_Revolut...	onto:Story

201 results

2. Выбрать все строения:

   Запрос:

   ```
   SELECT ?Subject WHERE {
   ?Subject a onto:Building. }
   ```

   Результат:

?Subject
onto:Place_of_building
onto:Rocket_Turret
onto:Citadel
onto:Power_Station
onto:Laser_Turret
onto:Barrack
onto:Mine

3. Выбрать все игровые сущности, стоимость постройки которых больше или равно 500:

   Запрос:

   ```
   SELECT ?Entity WHERE {
   ?Entity onto:UseCost ?cost. FILTER (?cost >= 500). }
   ```

   Результат:

?Entity
onto:Barrack
onto:Laser_Turret
onto:Rocket_Turret

4. Выбрать все игровые сущности, которые принадлежат команде игрока:

   Запрос:

   ```
   SELECT ?Entity ?Team WHERE {
   ?Entity onto:BelongsTeam ?Team.
   FILTER (?Team = onto:Player) }
   ```

   Результат:

?Entity	?Team
onto:Barrack	onto:Player
onto:Citadel	onto:Player
onto:Explosion	onto:Player
onto:Freeze	onto:Player
onto:Landing_Robot	onto:Player
onto:Laser_Turret	onto:Player
onto:Main_Robot	onto:Player
onto:Mine	onto:Player
onto:Place_of_building	onto:Player
onto:Rocket_Turret	onto:Player

5. Выбрать все игровые сущности с радиусом атаки больше или равной 3, принадлежащих команде противника:

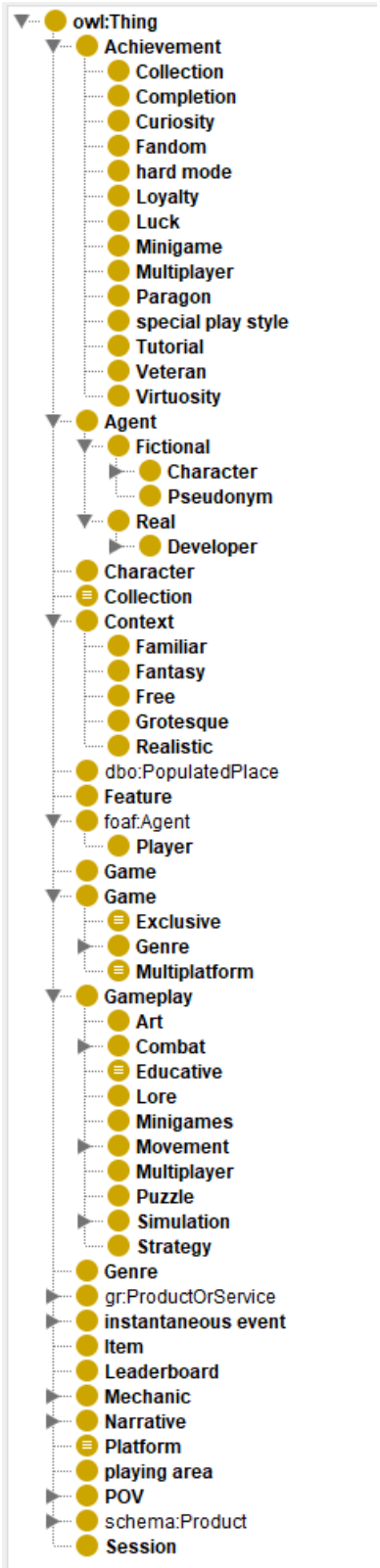
   Запрос:

```
SELECT ?Entity ?Team WHERE {
?Entity onto:AttackRadius ?range.
?Entity onto:BelongsTeam ?Team.
FILTER (?range >=3 && ?Team = onto:Enemy). }
```

   Результат:

?Entity	?Team
onto:Fast_Robot	onto:Enemy

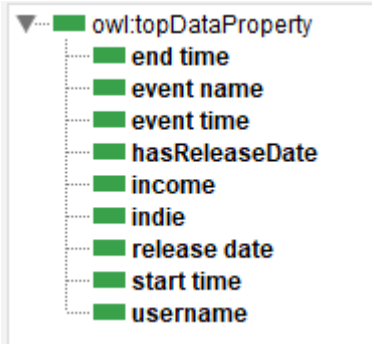
Для рассмотрения функционала объединения онтологий использовались онтологии из https://ceur-ws.org/Vol-3579/paper15.pdf и https://oa.upm.es/72411/1/REV_JCR_42_C.pdf



Они объединялись между собой, но не между разрабатываемой технологией по причине того, что они схожи по предметной области, но функционально они различаются. Структура классов объединенной онтологии выглядит следующим образом:

Она включает в себя следующие объектные отношения:

Объединённая онтология включает в себя следующие информационные свойства:



Запросы к данной онтологии:
Для начала следует отметить что для сокращения количества текста без потери информативности, во всех запросах использовался следующий набор импортов:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX VideOWL: <http://www.semanticweb.org/porta/ontologies/2022/5/VideOWL#>
PREFIX unt: <http://www.semanticweb.org/porta/ontologies/2022/5/untitled-ontology-7#>
PREFIX GameOnt: <http://purl.org/net/VideoGameOntology#>
```

1. Выбрать все игры серии Resident Evil:

   Запрос:

```
SELECT DISTINCT ?Developer
WHERE {
        ?Developer unt:developerOf ?Game.
        FILTER (?Game = unt:MINECRAFT) }
```

   Результат:

?Developer
unt:Markus_Perrson
unt:Mojang
unt:Notch

2. Выбрать все игры, являющиеся шутерами от первого лица и их доходы:

Запрос:

```
SELECT (COUNT(?Game) AS ?cnt) ?Game
WHERE {
        ?Game unt:compatibleWith ?o }
GROUP BY ?Game
```

Результат:

?cnt	?Game
1	unt:CALL_OF_DUTY
2	unt:FABLE
3	unt:MONUMENT_VALLEY_II
1	unt:BINDING_OF_ISAAC
3	unt:MINECRAFT
2	unt:ASSASSIN'SCREED
1	unt:BIG_BRAIN_ACADEMY

3. Выбрать количество платформ, поддерживаемых игровыми сериями:

Запрос:

```
SELECT ?Game
WHERE {
        ?Game unt:compatibleWith?Platform.
        FILTER(?Platform=unt:Pc_gaming) }
```

Результат:

?Game
unt:BINDING_OF_ISAAC
unt:CALL_OF_DUTY
unt:FABLE
unt:MINECRAFT
unt:MONUMENT_VALLEY_II

Protege: инструмент для создания онтологий

Protege – это бесплатный, открытый и многоплатформенный инструмент с графическим интерфейсом, предназначенный для создания, редактирования и управления онтологиями. Он используется в различных областях, включая искусственный интеллект, биоинформатику, медицину, образование и инженерию.

Возможности Protege:

- Создание онтологий с использованием различных языков онтологий, таких как OWL, RDF, RDFS и OBO.
- Редактирование онтологий с помощью интуитивно понятного графического интерфейса.
- Управление онтологиями, включая импорт, экспорт, версионирование и отладку.
- Визуализация онтологий с помощью различных диаграмм.
- Интеграция с другими инструментами и приложениями. Функционал Protege:
  - Редактор классов и экземпляров: позволяет создавать, редактировать и удалять классы, экземпляры и отношения между ними.
  - Редактор слотов: позволяет создавать, редактировать и удалять слоты, а также задавать их типы и ограничения.
  - Редактор правил: позволяет создавать, редактировать и удалять правила, которые определяют логику онтологии.
  - Модуль проверки: позволяет проверять онтологию на наличие ошибок и противоречий.
  - Модуль рассуждений: позволяет использовать онтологию для выполнения рассуждений и вывода новых знаний.

Достоинства Protege:

- Бесплатный и открытый: Protege можно использовать бесплатно и без ограничений.
- Многоплатформенный: Protege работает на Windows, macOS и Linux.
- Простота использования: Protege имеет интуитивно понятный графический интерфейс, который делает его доступным для пользователей с разным уровнем подготовки.
- Мощный функционал: Protege обладает богатым набором функций, которые позволяют создавать и управлять сложными онтологиями.
- Расширяемость: Protege можно расширить с помощью плагинов, которые добавляют новые функции и возможности.

Недостатки Protege:

- Крутая кривая обучения: Protege имеет множество функций, поэтому освоение всех его возможностей может занять некоторое время.
- Некоторые функции могут быть сложными: Некоторые функции Protege, такие как редактор правил, могут быть сложными для пользователей с ограниченным опытом работы с онтологиями.
- Отсутствие некоторых функций: Protege не хватает некоторых функций, которые есть в других инструментах для создания онтологий, например, возможности совместной работы.

Вывод

В ходе данной лабораторной работы были рассмотрены принципы создания онтологий. В ходе данной лабораторной работы был использован инструмент под названием Protege. С его помощью была создана онтология, а также проведена работа по объединению и использованию существующих онтологий.