Capturing Game Telemetry with Provenance

SBGames 2017 Curitiba PR

Autores: **Troy C. Kohwalter,** Leonardo G. P. Murta, Esteban W. G. Clua

{tkohwalter, leomurta, esteban}@ic.uff.br

Instituto de Computação – Universidade Federal Fluminense (UFF)

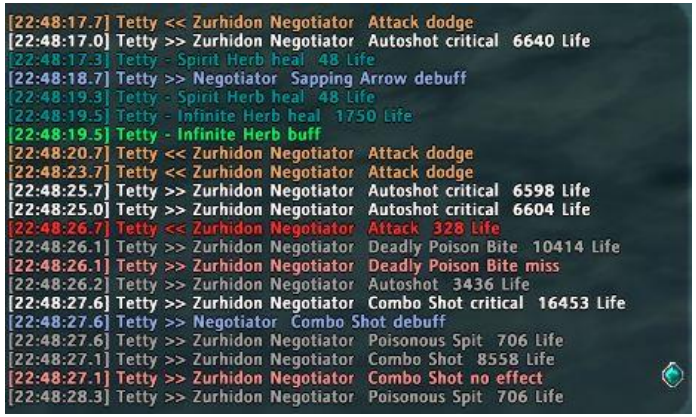
Instituto de Computação

MEDIALAB

ESI Laboratório de Engenharia de Sistemas e Informação

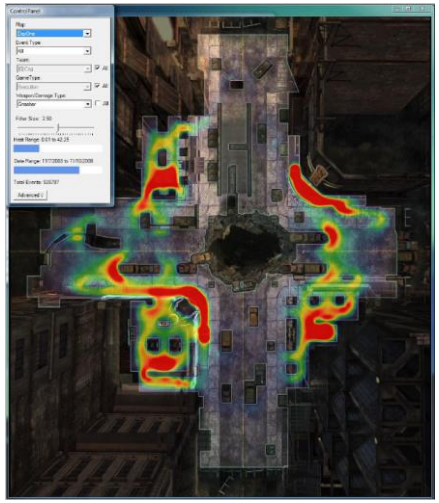
Game Telemetry

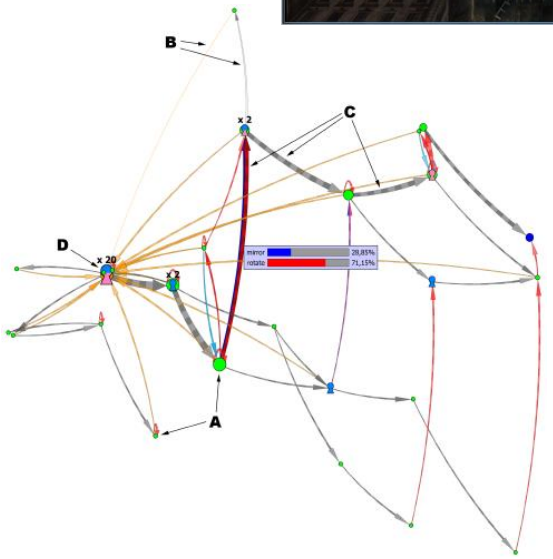
- Telemetry
  - Data gathered over distance
- Data gathering
  - Important aspect in the industry
  - Game development, research and production
- Advantages
  - Stability
  - Dynamic balancing
  - Behavior analysis
  - Monetization
  - Detect problems



Game Telemetry Gathering

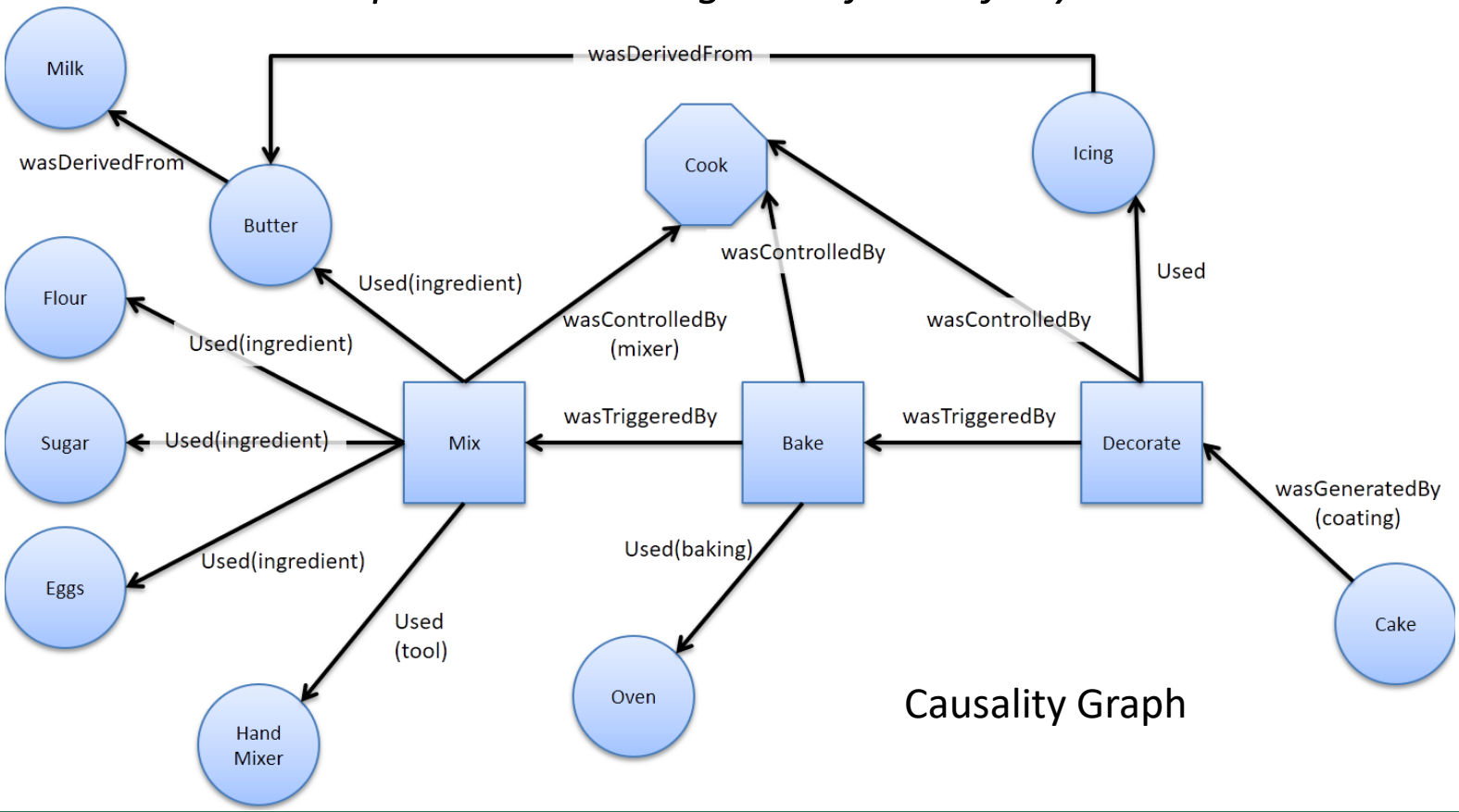
- Challenging
  - Game complexity
  - Information size
- What do we track?
  - State changes *(Y.-E. Liu et. Al., 2011), (G. Wallner, 2013), (M. S. El-Nasr and T.-H. Nguyen, 2015)*
    - Coarse grain
  - Events *(S. Joslin et al., 2007), (J. H. Kim et al., 2008)*
    - Fine grain
- Cause-and-effect relationships
  - A.K.A. influences
  - **Not captured!**





Provenance

- *"Refers to the documented history of an art object, or the documentation of processes in a digital object's life cycle"* http://www.w3.org/TR/prov-primer/



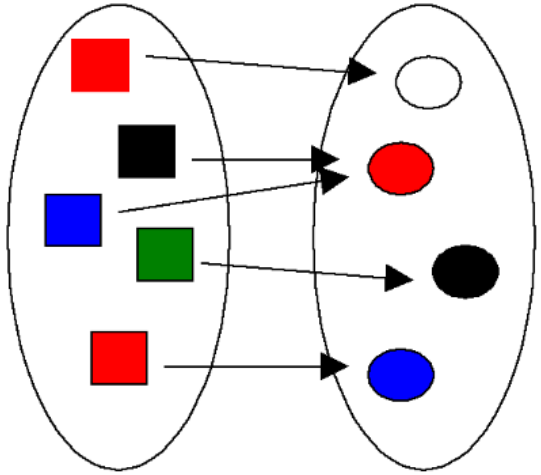
Provenance in Games (PinG)

Conceptual framework

(Kohwalter et al., 2012)

- First work in the field

- Domain mapping
  - Brings provenance to games



- Tracking and gathering
  - Provenance data
  - Causal relationships

PinG

(Kohwalter et al., 2012)

- Entity
  - Objects
- Activity
  - Actions
  - Events
- Agent
  - NPCs
  - Player

PinG

(Kohwalter et al., 2012)

- Entity
  - Objects
- Activity
  - Actions
  - Events
- Agent
  - NPCs
  - Player

PinG

(Kohwalter et al., 2012)

- Entity
  - Objects
- Activity
  - Actions
  - Events
- Agent
  - NPCs
  - Player






Goals

- Capture **causal relationships** telemetry
  - Based on PinG's conceptual framework
- Easy integration
  - PinG had no actual implementation
  - Minimum coding
    - Automatic information management

Car Tutorial

https://www.assetstore.unity3d.com/en/#!/content/10

- Capture provenance telemetry from game session
- Generate provenance graph

VIDEO

Grafo gerado no *Prov Viewer (Kohwalter et al., 2016)*
http://gems-uff.github.io/prov-viewer/

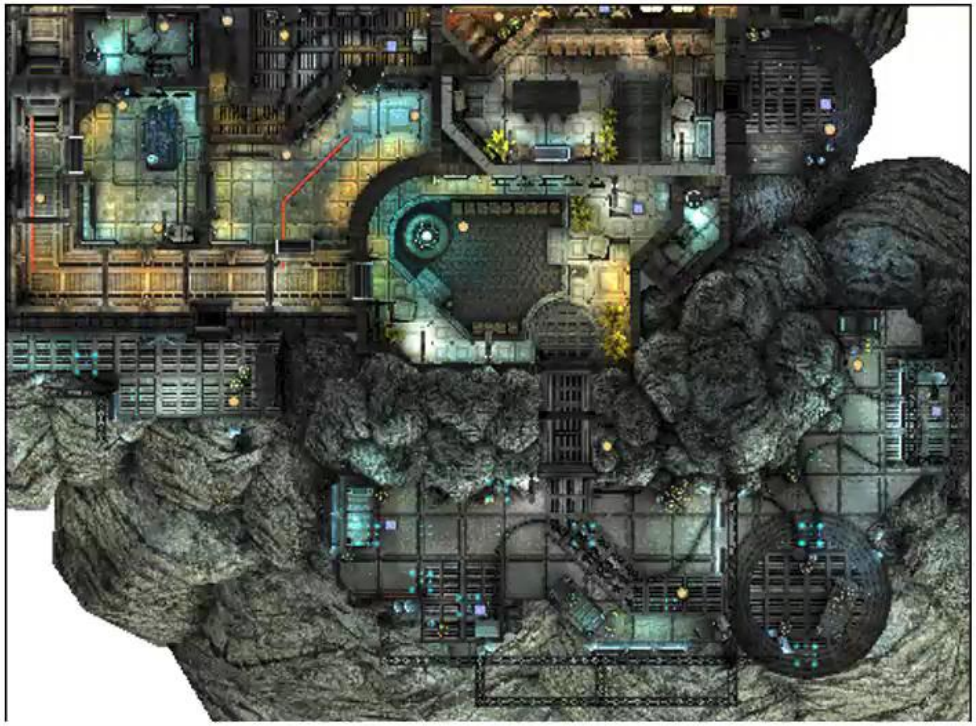
Angry Bots

https://www.assetstore.unity3d.com/en/#!/content/12175

- Capture provenance telemetry from game session
- Generate provenance graph

VIDEO

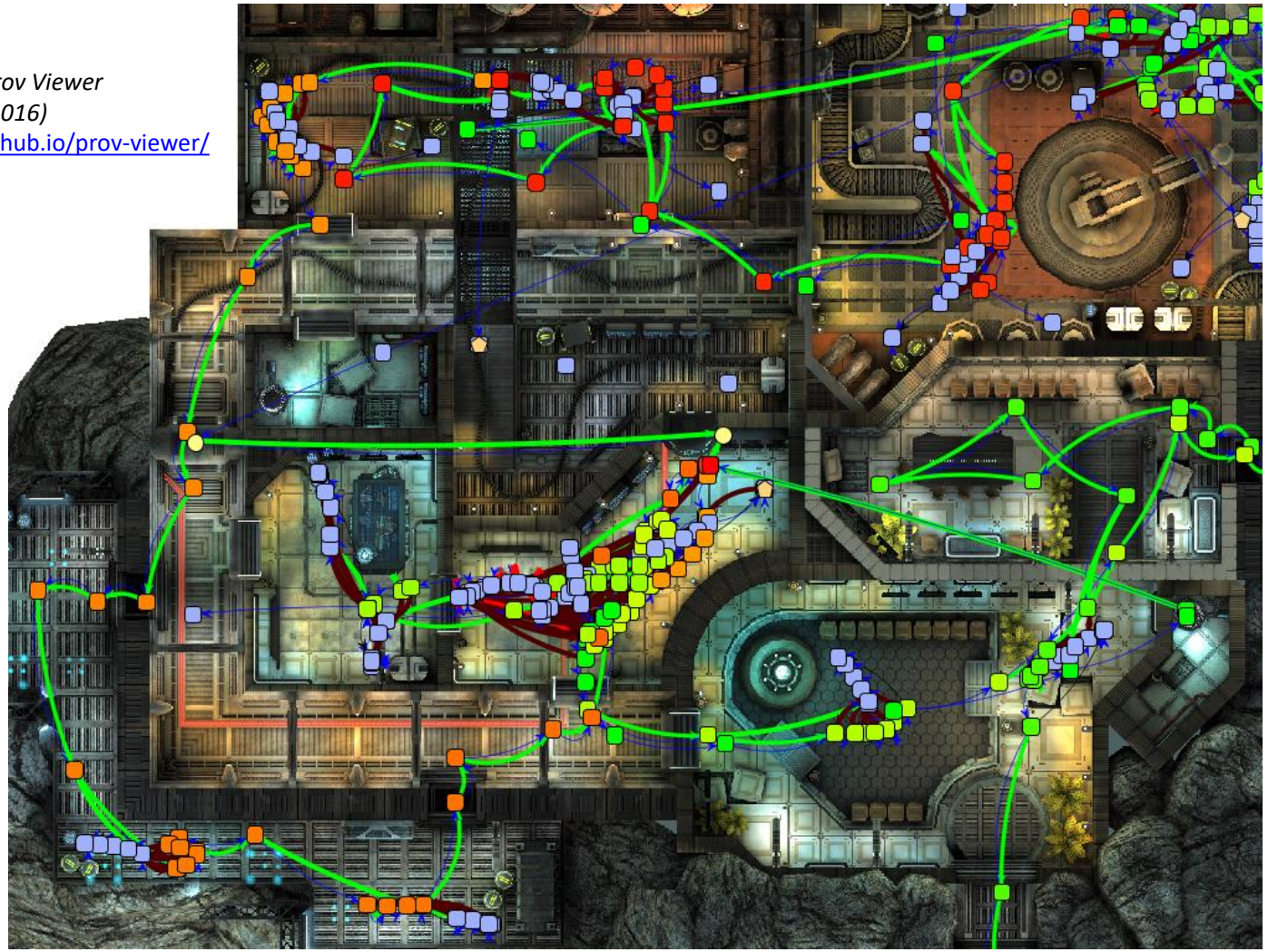
Grafo gerado no *Prov Viewer (Kohwalter et al., 2016)*
http://gems-uff.github.io/prov-viewer/



Angry Bots

https://www.assetstore.unity3d.com/en/#!/content/12175

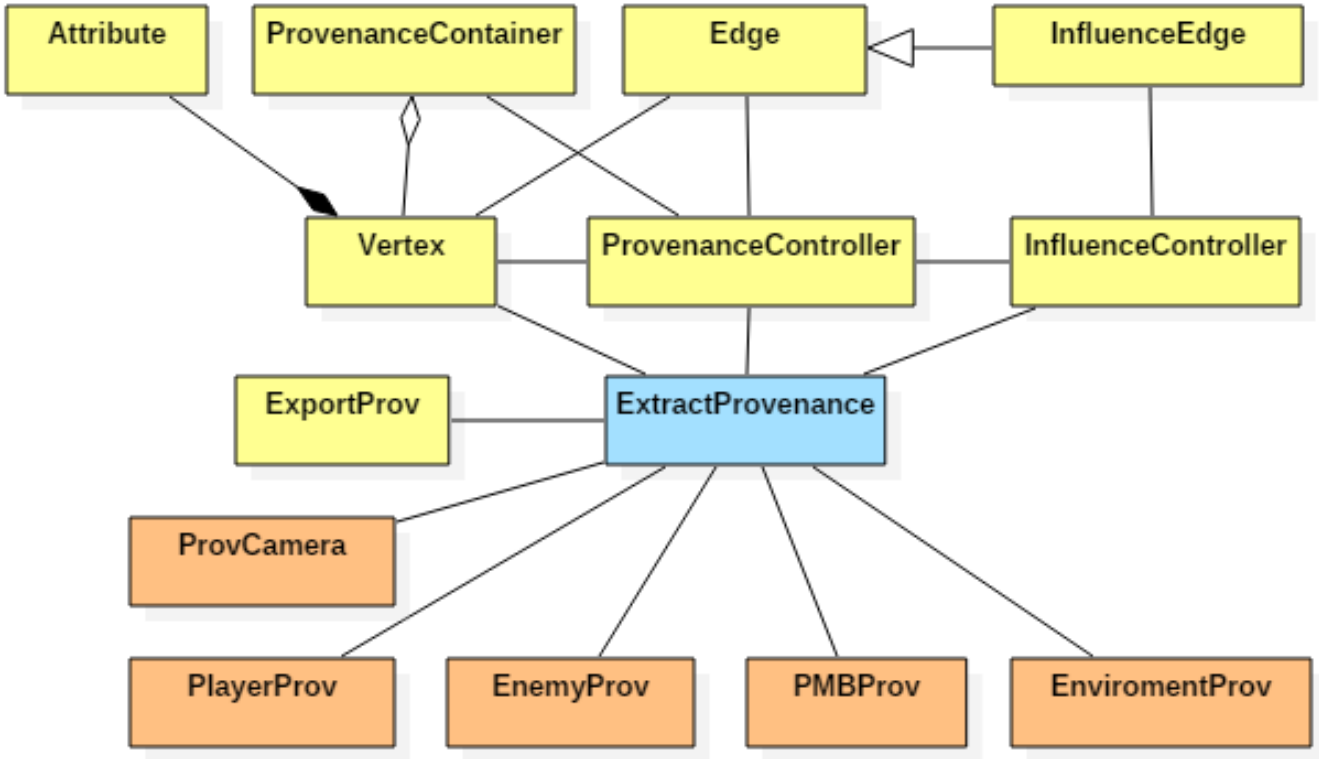
Grafo gerado no *Prov Viewer*
(Kohwalter et al., 2016)
http://gems-uff.github.io/prov-viewer/



HOW?

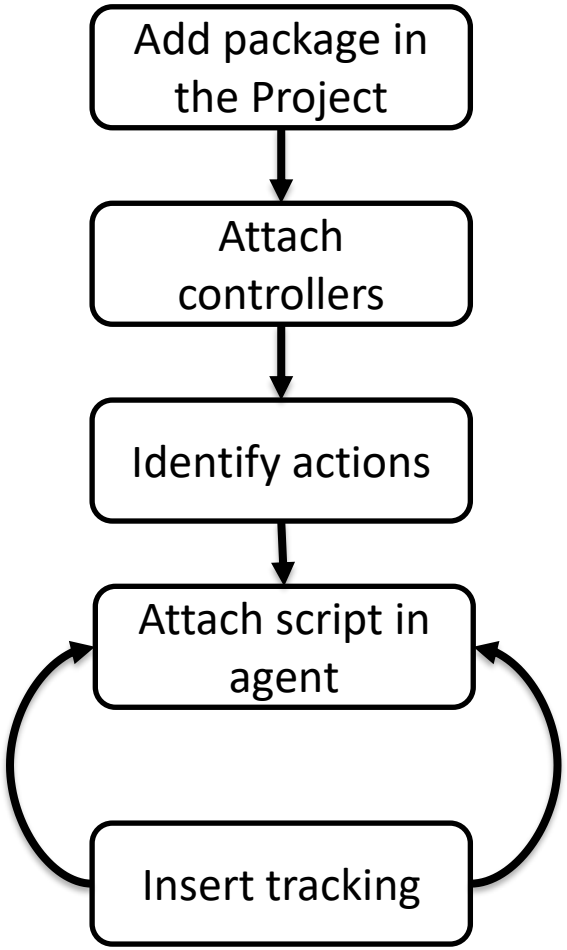
Provenance in Games for Unity - PinGU

- Yellow classes
  - Management
- Blue class
  - Gatherer
- Orange classes
  - Domain-Specific



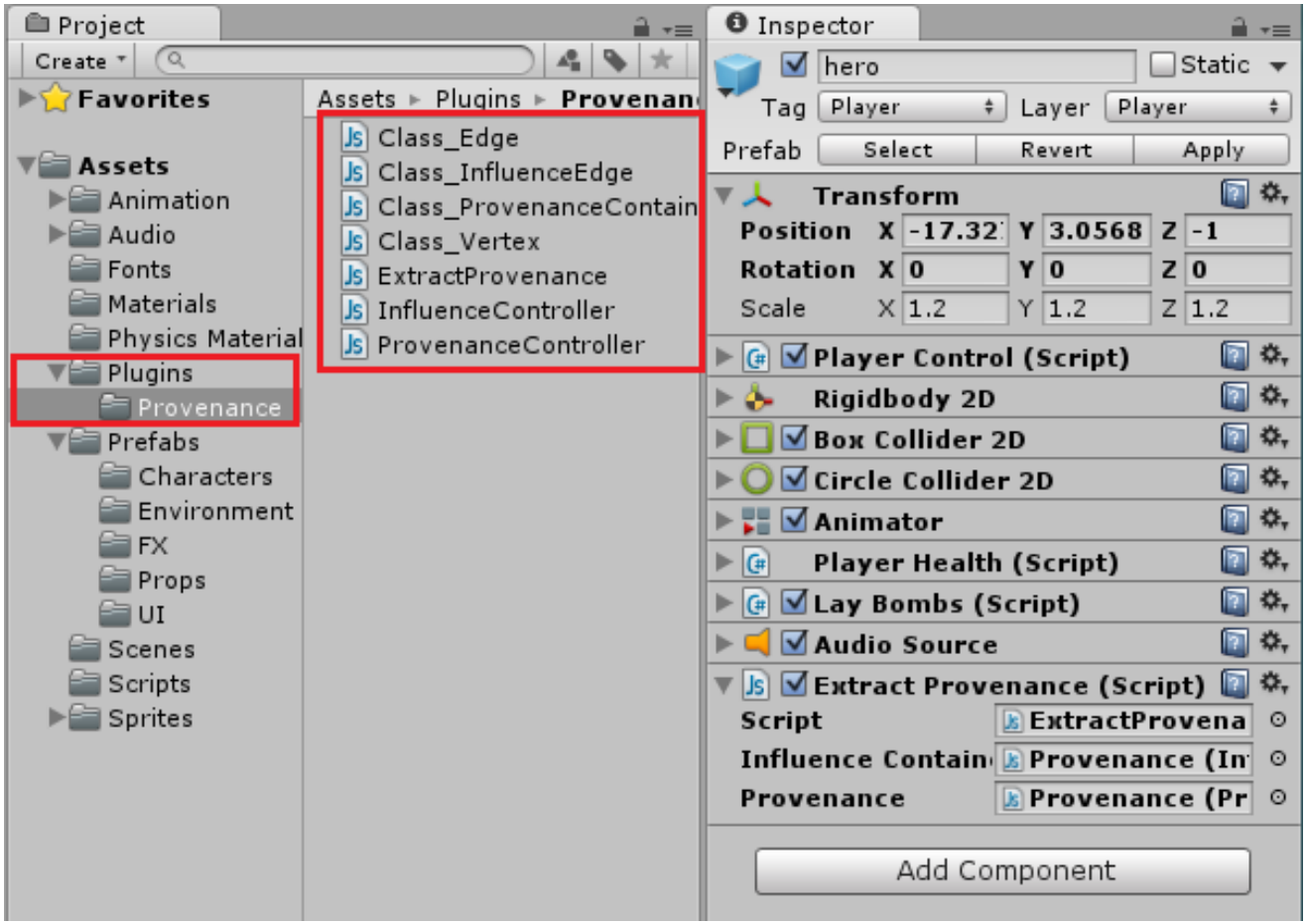
Integration

1. Import PinGU
2. Provenance controllers
    1. *Provenance Controller*
    2. *Influence Controller*
3. Game Design
    1. Identify actions
    2. Identify scripts
4. Provenance extractor
    1. *Extract Provenance*
5. Provenance tracking functions
    1. Domain specific
    2. Insert in existing scripts



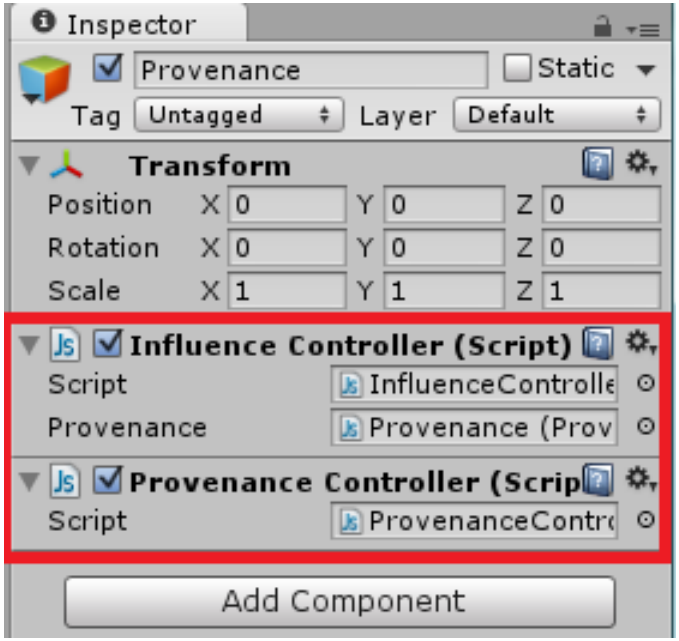
Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts



Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts



Integration

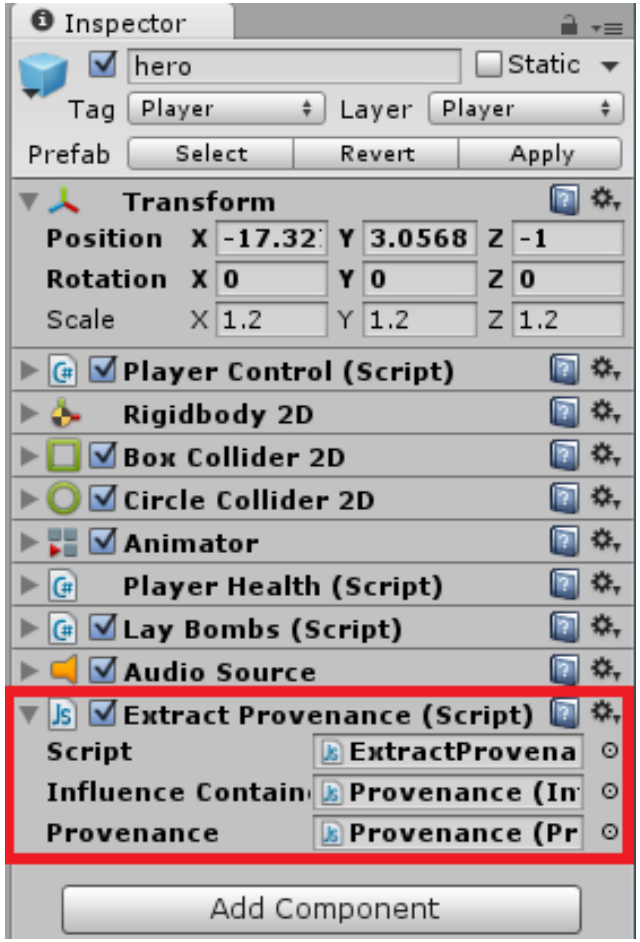
1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts

Game design summary (example)

- **Enemy**
  - –Health (Attribute)
  - –Enemy Attack (Action)
    - •Player take Damage (Influence)
- **Player**
  - –Health (Attribute)
  - –Throw Bomb
    - •Secondary Attack (Action)
    - •Damage Enemy, Area of Effect (Influence)
  - –Fire Gun
    - •Primary Attack (Action)
    - •Spawn projectile
  - –Projectile
    - •Enemy Damage (Influence)
  - –Player Death (Action)
- **Item Spawner**
  - –Create Health Item (Action)
    - •Health Item (Object)
    - •Heal (Influence)
  - –Create Bomb Item (Action)
    - •Bomb container (Object)
    - •Increase Bomb ammunition (Influence)

Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts



Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts

Tracking Function insertion

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;



    }
}
```

Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts

Tracking Function insertion

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

Domain-specific tracking function

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts

Domain attributes

Tracking Function insertion

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

Domain-specific tracking function

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts

Information Type

Tracking Function insertion

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

Domain-specific tracking function

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Integration

1. Import PinGU
2. Provenance Controllers
   1. *Provenance Controller*
   2. *Influence Controller*
3. Game Design
   1. Identify actions
   2. Identify scripts
4. Provenance extractor
   1. *Extract Provenance*
5. Tracking functions
   1. Domain specific
   2. Insert in existing scripts

Causal Relationship

Tracking Function insertion

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

Domain-specific tracking function

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Contributions

- Capturing provenance data
  - Data exploration
  - Behavior analysis
  - Other abstract analysis

- **Casual relationships**
  - Cause-and-Effect

Future work

- Visualization
- More automatization during integration
- Dynamic balancing
- Modeling provenance graph for Deep Learning

Capturing Game Telemetry with Provenance

SBGames 2017 Curitiba PR

PinGU
http://gems-uff.github.io/ping/

Apresentação
SBGames 2017

Prov Viewer
http://gems-uff.github.io/prov-viewer/

Autores: **Troy C. Kohwalter,** Leonardo G. P. Murta, Esteban W. G. Clua

{tkohwalter, leomurta, esteban}@ic.uff.br

Instituto de Computação – Universidade Federal Fluminense (UFF)

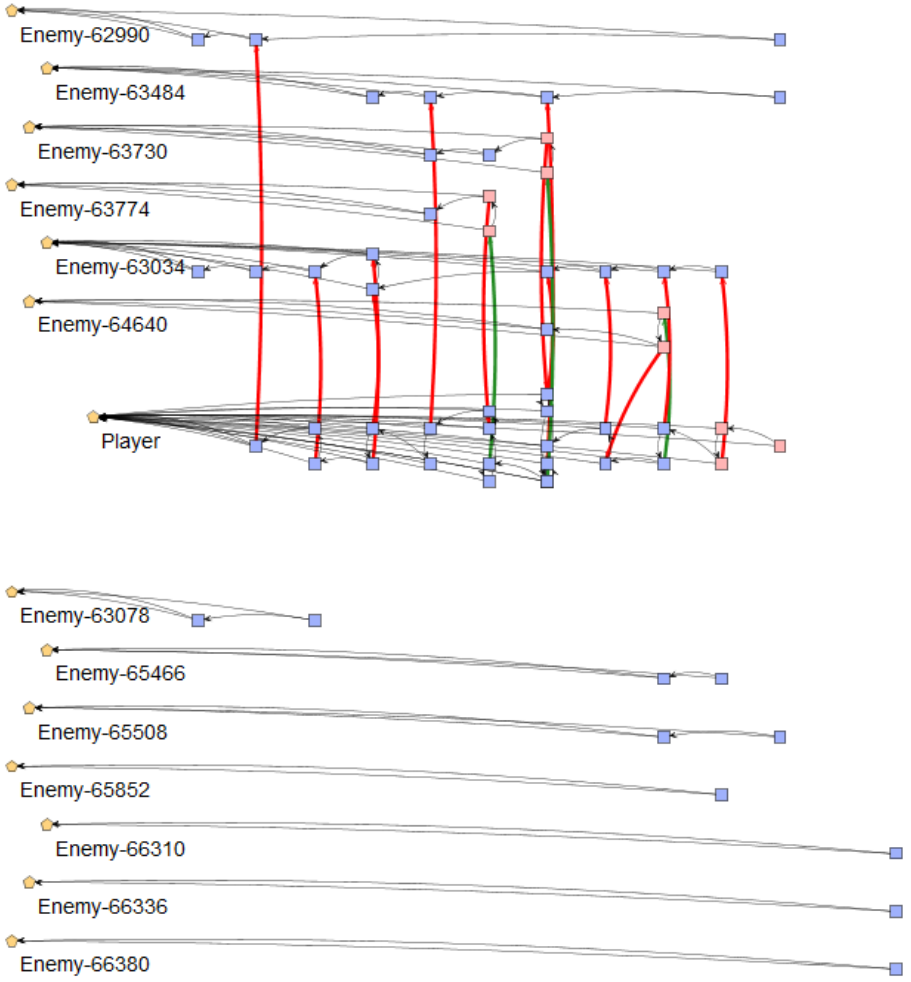
Instituto de Computação

MEDIALAB

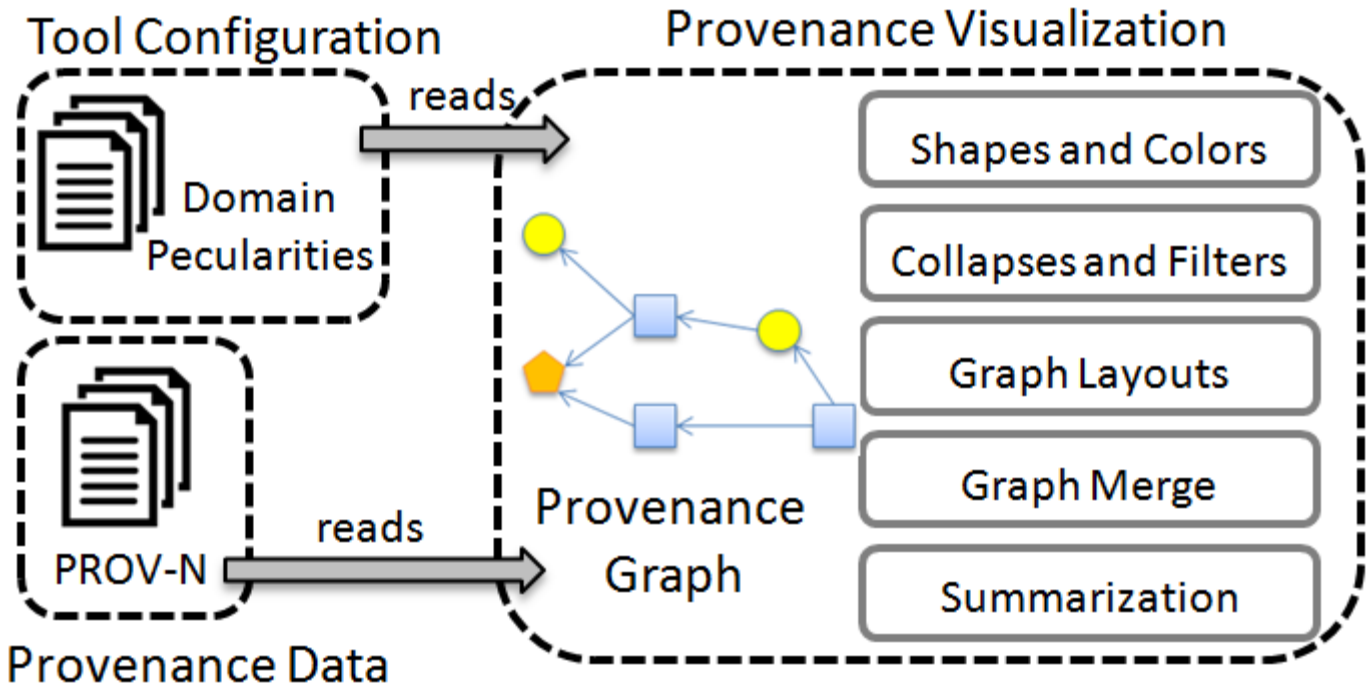
ESI Laboratório de Engenharia de Sistemas e Informação

Provenance Graph

- All agents
- Executed actions
  - Who executed
  - Causal-Relationships
    (Vertical edges)



Prov Viewer


Tool Configuration
Provenance Visualization
reads
Domain
Pecularities
Shapes and Colors
Collapses and Filters
Graph Layouts
Graph Merge
Summarization
Provenance
Graph
PROV-N
reads
Provenance Data