REWIRE

Computational Intelligence in Games

Michele Pirovano
Research Assistant at UNIMI, AisLAB
PhD Student at POLIMI
Game Design, Artificial Intelligence and Games for Health

REWIRE

What is Computational Intelligence?

- *«**Computational intelligence (CI)** is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which traditional approaches are ineffective or infeasible»*
- In short terms: practical solutions!
- Examples:
  - Fuzzy Logic
  - Artificial Neural Networks
  - Evolutionary Computation

REWIRE

Schedule

- The REWIRE project and IGER
- Computational Intelligence for Rehab Games
  - Fuzzy Logic for Monitoring
  - Bayesian Adaptation
  - Procedural Content Generation (PCG)
- Computational Intelligence for Games
  - PCG for Games
  - Fuzzy Logic (Fuzzy Tactics)
  - Reinforcement Learning Agents
  - Evolutionary Computation
  - Etc.
- Bonus Topic: Kinect Fusion

REWIRE

Computational Intelligence in REWIRE

The REWIRE Project

- Goal
  - Bring rehabilitation to the patient's home
- Hierarchy
  - Networking station
  - Hospital station
  - Patient station

Games & Rehab

- Exergames: games + exercises
  - Games can be useful to provide *fun*
  - Exercises are required for rehab efficacy

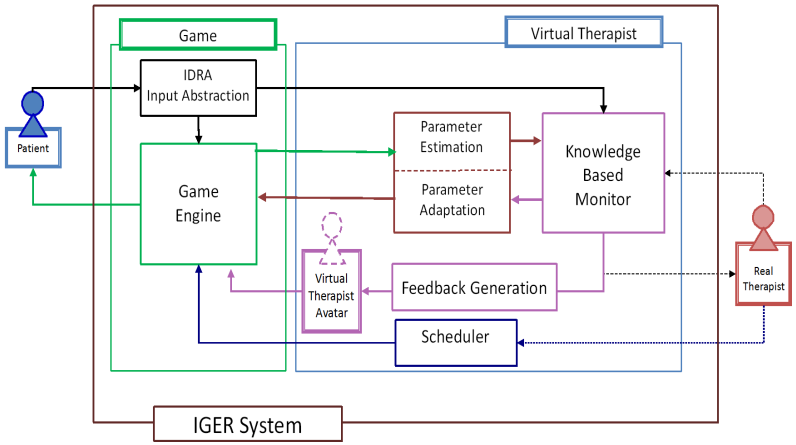
Commercial Games
lack the needed features for rehabilitation

Virtual Reality Rehab
is not done following good game design practices

- Keep in mind: we are at home, so there is no therapist!

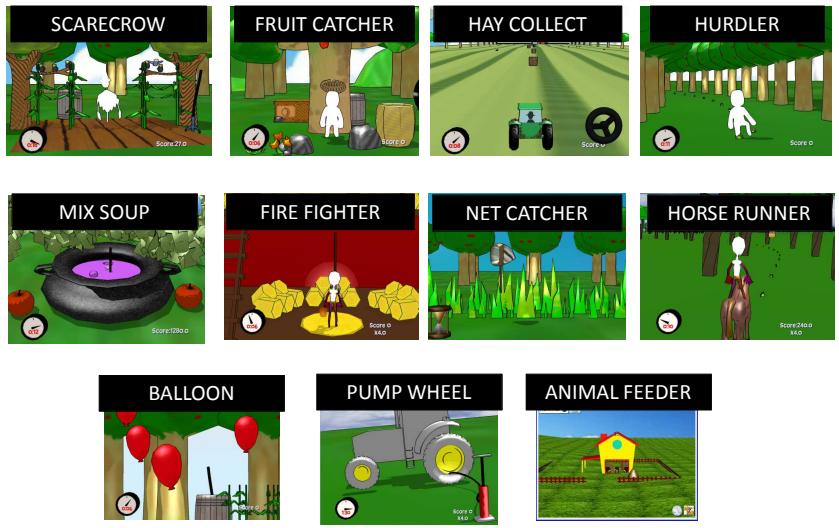
REWIRE

Intelligent Game Engine for Rehabilitation: IGER



REWIRE

Our Games



REWIRE

Features for Rehabilitation Games

Efficacy

- Configuration
- Recording
- Monitoring

Accessibility

- Multiple Device Support
- Natural User Interfaces
- Adaptation

Motivation

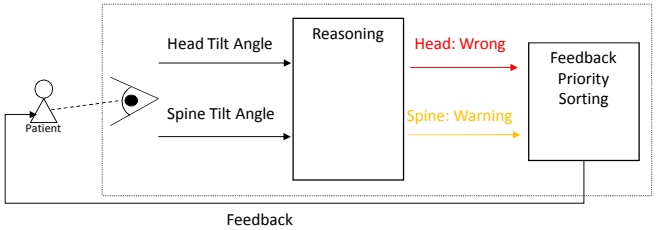
- Virtual Therapist Avatar
- Controlled Randomization
- Reward System

REWIRE

Monitoring

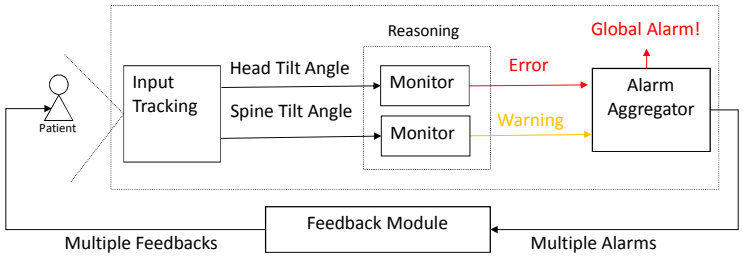
- **Goal:** On-line monitoring and correction of the user's movements
  - Major feature that must be provided even in absence of the therapist

How the therapist monitors…



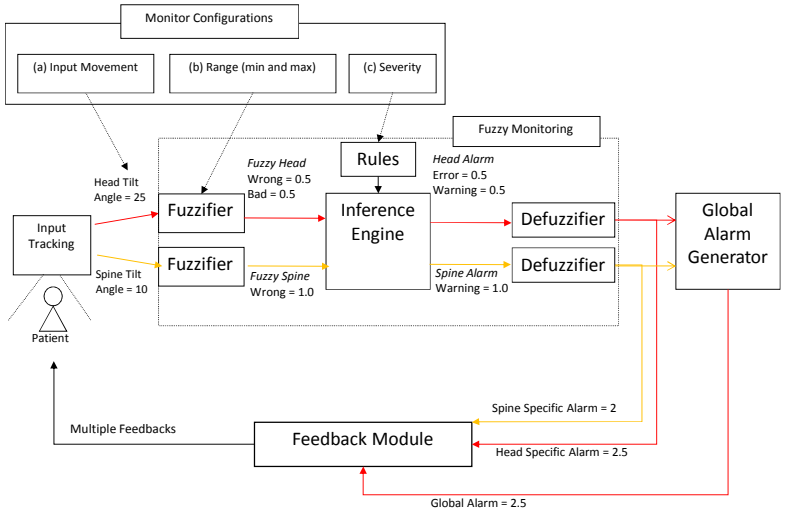
- Feedback is either verbal or physical. But we cannot touch the patient!
- We can however separate different 'body parts' and treat them disjointly…

Monitoring by our Virtual Therapist



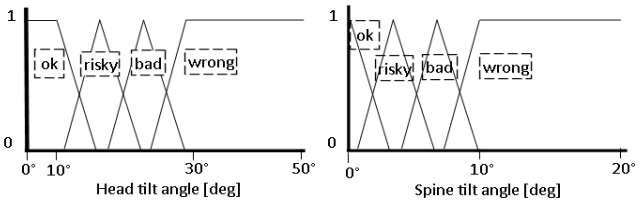
- Now, we need to create the monitors so that:
  - They encode the therapist's knowledge
  - They can be easily configured by therapists
  - They are robust to uncertainity

Monitoring through Fuzzy Logic

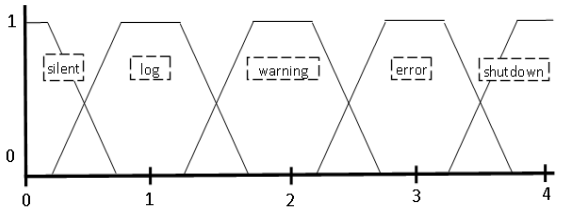


Fuzzy Monitoring in detail

- Tracked Input is encoded as the Fuzzy Input



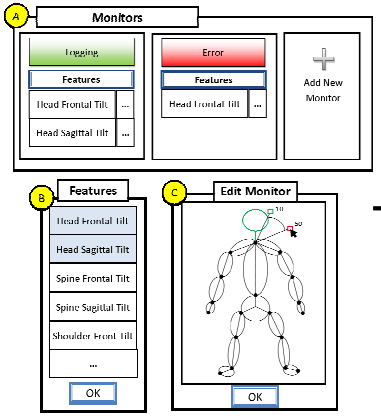
- The Alarm Level is chosen as the Fuzzy Output



REWIRE

Fuzzy Monitoring in detail

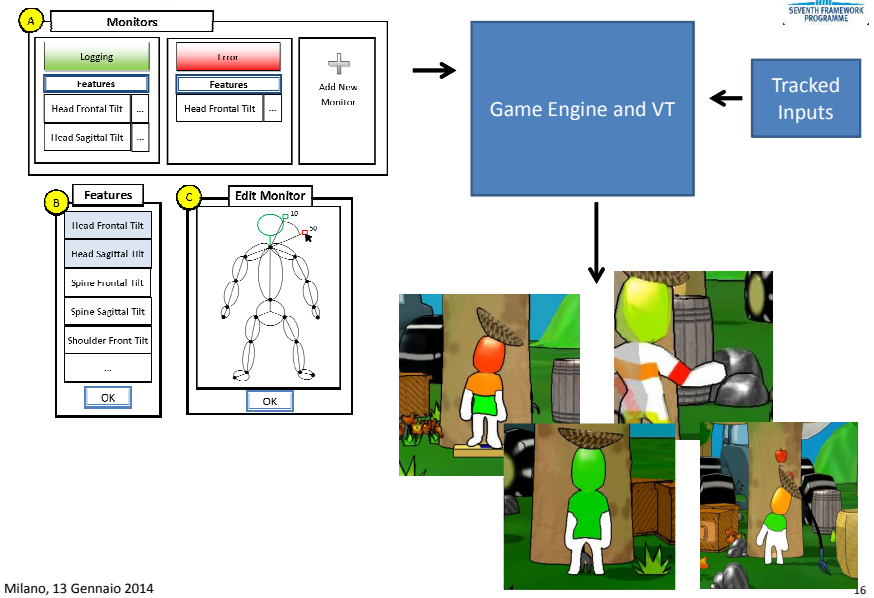
- The Fuzzy Rules are generated automatically based on the monitors defined by the therapist



If HEAD TILT is OK, then ALARM is SILENT
If HEAD TILT is RISKY, then ALARM is LOG
If HEAD TILT is BAD, then ALARM is WARNING
If HEAD TILT is WRONG, then ALARM is ERROR

REWIRE

Monitoring: result



REWIRE

Monitoring: video

REWIRE

Dynamic Difficulty Adaptation

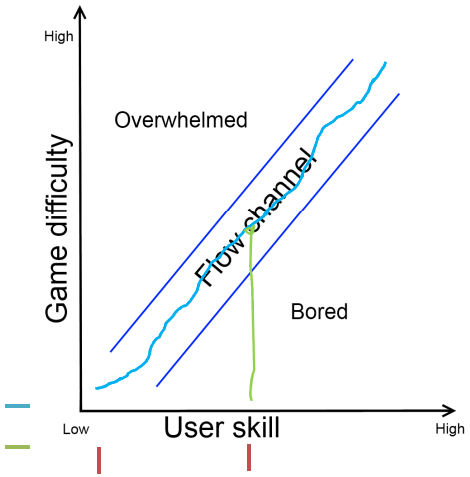
- **Goal:** adapt the game to the capabilities of the user, maintaining challenge and avoiding frustration
- We want the games to be playable by people with very diverse conditions

- This is done even in entertainment games, but it is much more critical for us!

REWIRE

Theory of Flow

- **Good game design**
- **With DDA**



REWIRE

DDA concepts

- Performance ***P***
  - *We estimate how good or how badly the player is doing*
- Adapted parameter ***x***
  - *This is directly related to the difficulty of the game*
  - *Examples: enemy health in Oblivion, accuracy correction in Max Payne*
- Adaptation delta ***dx***
  - *This is the amount of change of the adapted parameter*
- *At each adaptation step:* ***x = x + dx***
- *And:* ***dx= f(P)***

DDA methods

- Simple heuristics
  - Chose an estimation period (every N trials)
  - At each end of a period estimate the performance as number of hits on the total trials:

    $P = N_{success}/N_{tot}$
  - Adapt the parameter based on a performance threshold *T*

    dx = f(P) (example: P/100)

    if P > T: x += dx

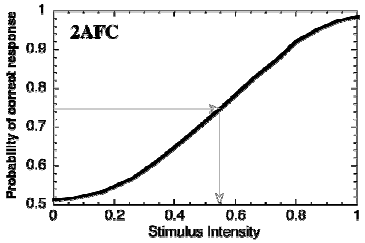
    if P < T: x -= dx

Extending DDA with CI

- What can we do to increase DDA efficacy?
  - We want any parameter to be adapted regardless of its actual function
  - ***The adaptation function should take into account previous history***
- Our approach: use the QUEST Bayesian method to estimate the player's skill from previous trials and converge to an optimum

QUEST concepts

- *QUEST: a Bayesian adaptive psychometric method*
  [Watson&Pelli 1983]



- Psychometric function
  - Given a stimulus intensity, estimate the psychophysical response
  - Example:
    - flash a light in front of a person with different light intensity
    - check whether the person can see the light
    - after a few iterations, we can estimate the intensity threshold at which the light can be seen

QUEST concepts

- QUEST uses Bayes theorem to adapt the psychometric threshold of a stimulus following a set of trials
- Bayes theorem:

$$f_B(B)f_{A|B}(A|B) = f_A(A)f_{B|A}(B|A)$$

- For DDA, we define:
  - The outcome of the previous trials as the data *D* and its prior pdf as $f_D(D)$
  - The adapted parameter value *x* and its prior pdf $f_x(x)$ (an initial guess)
  - The posterior pdf $f_{x|D}(x|D)$ that will converge to the desired performance P_{end}
  - **The parameter $\mathbf{x_{end}}$, that is the value at which the performance of the patient is the chosen one, as the mean of $\boldsymbol{f_{x|D}(x|D)}$**

REWIRE

QUEST method

- Bayes theorem on our data:

$$f_{x|D}(x|D) = \frac{f_x(x) f_{D|x}(D|x)}{f_D(D)}$$

- This can be rewritten by taking the logarithm as:

$$Q(x) = \ln(f_x(x)) + \ln(f_{D|x}(D|x))$$

This is the probability distribution of the OUTCOMES given the PARAMETERS

- And by introducing the psychometric function $p_{r_i|x}(x_i)$

$$Q(x) = \ln(f_x(x)) + \ln(\prod_{i=1}^{n} p_{r_i|x}(x_i))$$

- $p_{r_i|x}(x_i)$ can be seen as $\Psi(-x_i)$

$$S(x_i) = \Psi(-x_i)$$
$$F(x_i) = 1 - \Psi(-x_i)$$

- And all of this can be rewritten as recursive

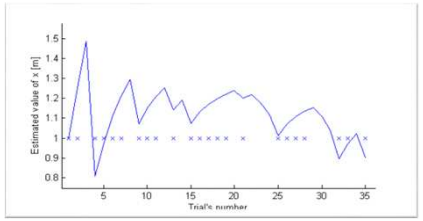
$$Q_i(x) = Q_{i-1}(x) + \begin{cases} S(x - x_i) & if\ success \\ F(x - x_i) & if\ failure \end{cases}$$

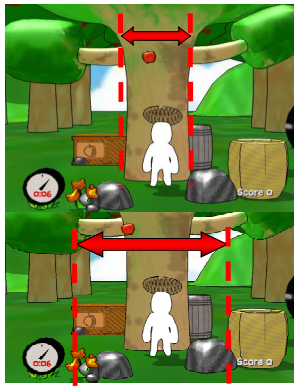
- At each new trial, the parameter value x is taken as the mean of $Q_i(x)$

REWIRE

Results

- The parameter gets adapted and converges to an optimal value after a few trials





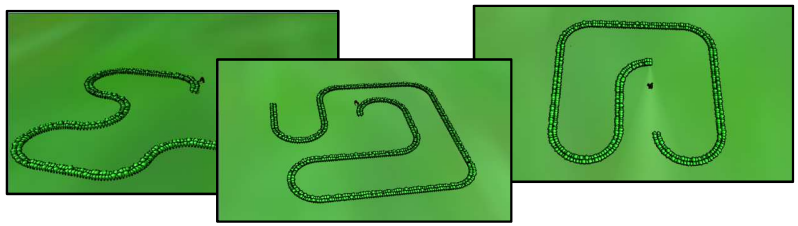
REWIRE

Adaptation Video

REWIRE

Controlled Randomization

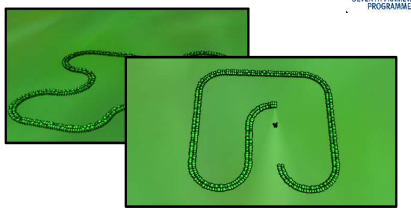
- Remember that the patients will have to perform the same exercise many, many times!
- Even if the games are fun, they would be come boring pretty soon
- **Goal:** introduce small changes to the games to make them more variable and thus more interesting



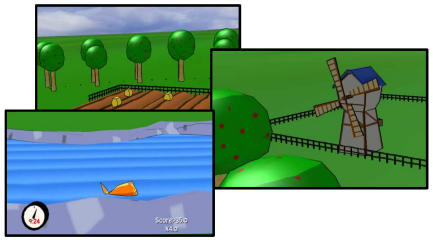
REWIRE

What do we randomize?

- Gameplay
  - Movement patterns
  - Gameplay parameters



- Graphics
  - Placement of *props*
  - Colors and textures



- Music
  - Randomized choice of background music

REWIRE

Procedural Content Generation

- We can go further with the whole «randomizing» thing: we can procedurally generate actual assets from scratch!
  - *"Procedural content generation (PCG) is the programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible game play spaces."* - http://pcg.wikidot.com/

REWIRE

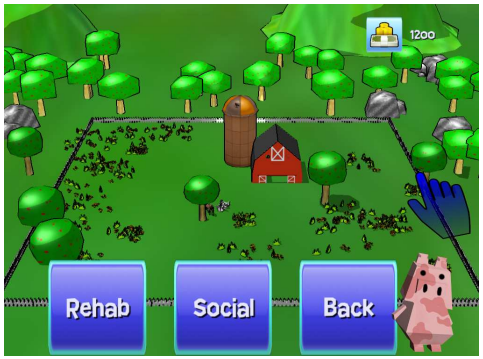
PCG in Rewire

- A first PCG method is already used for generating the paths
  - Starts from one end
  - Travels randomly
  - Avoids self-collisions

REWIRE

PCG in Rewire

- For the reward system of REWIRE, we aim to let the patient build a farm out of generated assets



REWIRE

PCG in Rewire

- We generate plants using L-systems:
  - A Lindenmayer system is a parallel rewriting system and a type of formal grammar
  - It possess a high degree of self-similarity
    - Like musical scores or like plants!
- Axiom: ***a***
- Production: ***a -> ab***
- Iterations:
  - ***a***
  - ***ab***
  - ***abb***
  - ***abbb***

REWIRE

PCG in Rewire

- Results:

REWIRE

PCG in Rewire: Video

SEVENTH FRAMEWORK PROGRAMME

REWIRE

SEVENTH FRAMEWORK PROGRAMME

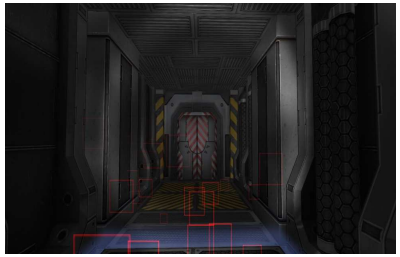
Other Applications of CI in games

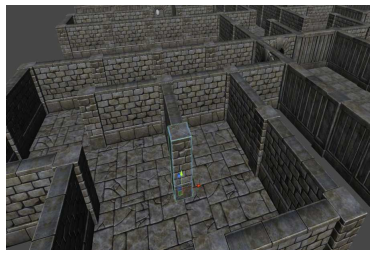
REWIRE Procedural Content Generation

- In commercial games:

REWIRE PCG: Dungeon Generator

- Recursive Backtracker algorithm generates a maze
- Specific algorithm generates rooms
- Parameterized for ease of use



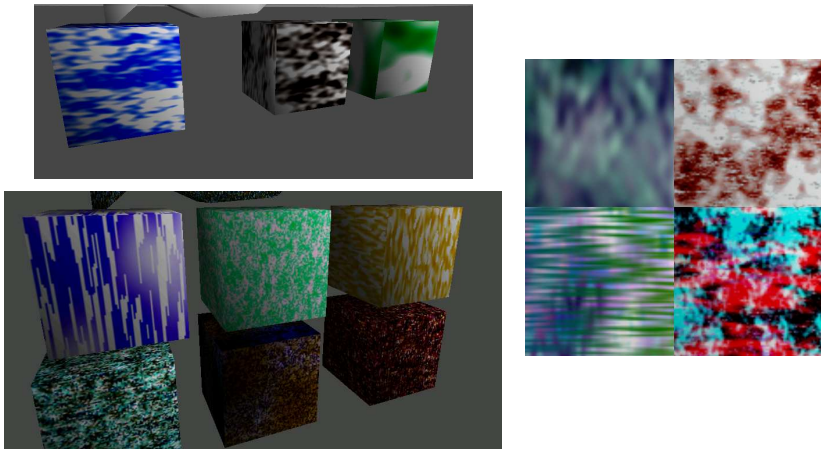


https://www.youtube.com/watch?v=A3174Wa8Ihk

REWIRE

PCG: Textures

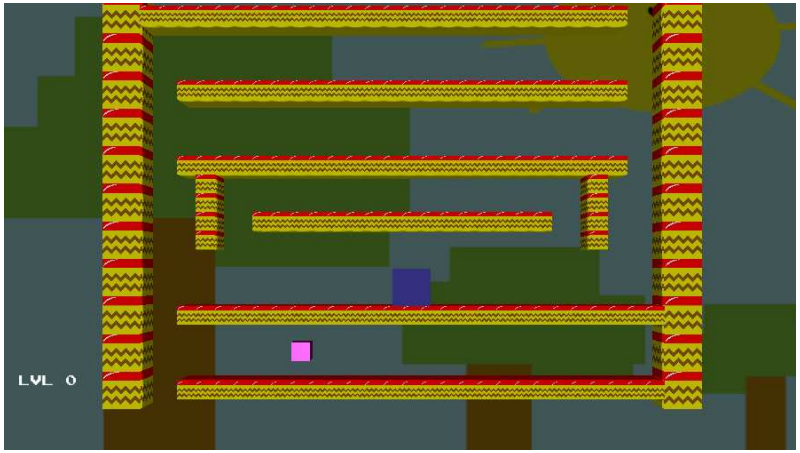
- Layers of simplex noise



REWIRE

PCG: Level design

- Randomized according to difficulty



http://www.indiedb.com/games/bulla-bolla

REWIRE

Fuzzy Logic

SEVENTH FRAMEWORK PROGRAMME

- In commercial games:

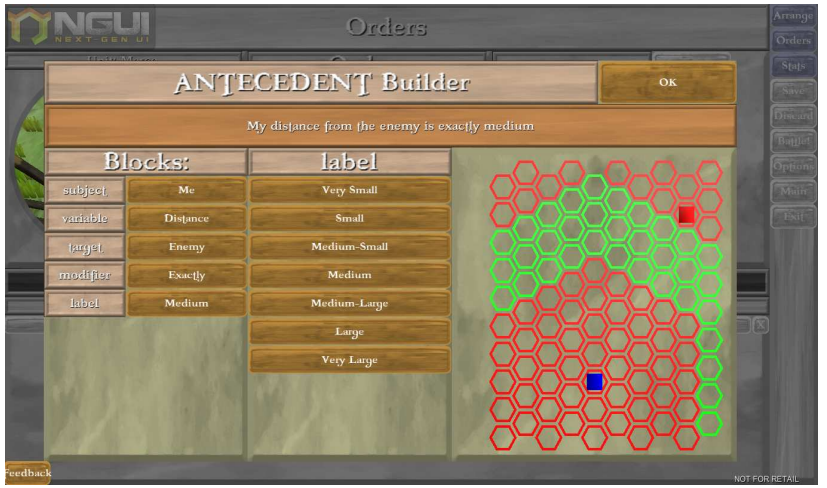
REWIRE

Fuzzy Tactics

SEVENTH FRAMEWORK PROGRAMME

Fuzzy Tactics: Logic Building

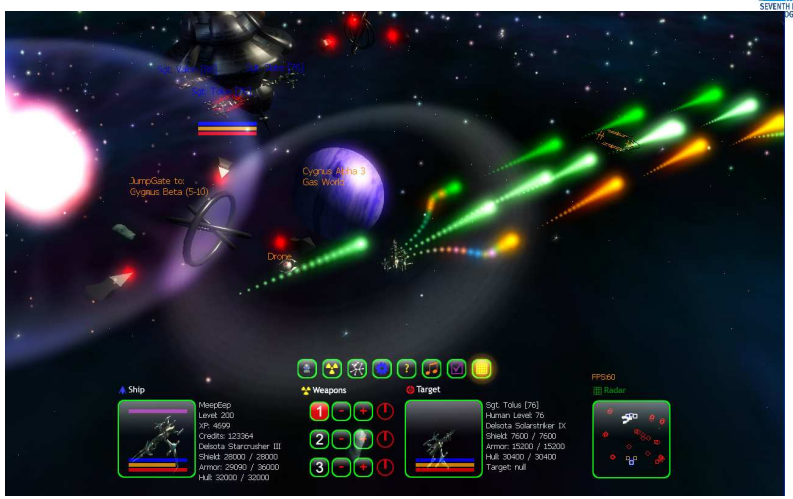
Fuzzy Tactics: Logic Building



Fuzzy Tactics: Video

https://www.youtube.com/watch?v=ft0PNrO2yEg#t=0

Evolutionary Computation



Hastings 2012 - Evolving Content in the Galactic Arms Race Video Game

REWIRE

Evolutionary Computation: Video

REWIRE

Reinforcement Learning

http://homes.di.unimi.it/~pirovano/games/RLogue/RLogue.html

Reinforcement Learning: Video

Kinect Fusion

Kinect Fusion is an algorithm developed by Microsoft Research in 2011. The algorithm allows a user to reconstruct a 3D scene in real-time and robustly by moving the Microsoft Kinect sensor around the real scene.

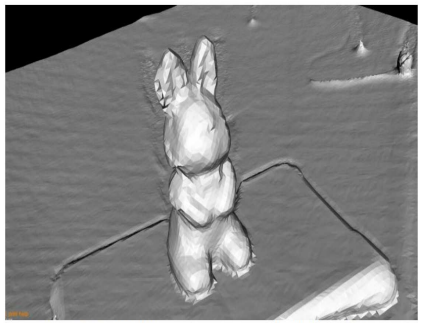


Figure 1 - A rabbit-like statue reconstructed with Kinect Fusion

http://homes.di.unimi.it/~pirovano/pdf/3d-scanning-pcl.pdf

REWIRE

Kinect Fusion: Algorithm

- Step 1: Get the depth images from Kinect

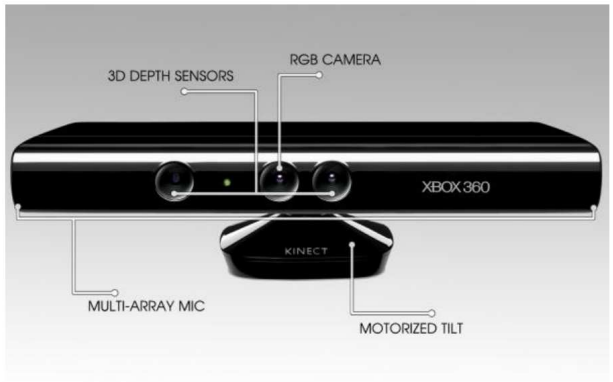


Figure 3 - The Microsoft Kinect sensor

REWIRE

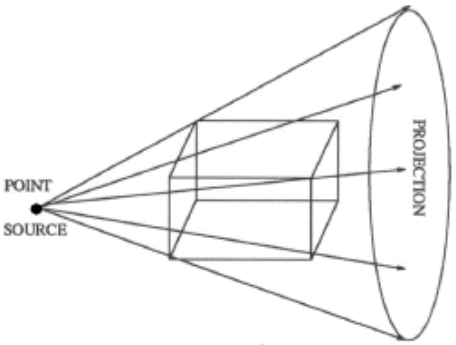
Kinect Fusion: Algorithm

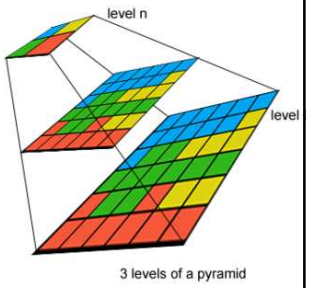
- Step 2: Apply a bilateral filter to the images

REWIRE

Kinect Fusion: Algorithm

- Step 3: Create a multi-resolution pyramid of 3D clouds through back-projection
  - With vertex and normal information

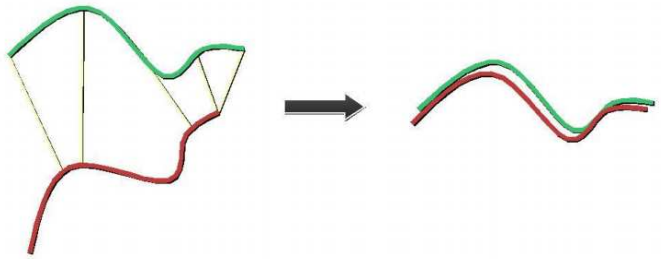




REWIRE

Kinect Fusion: Algorithm

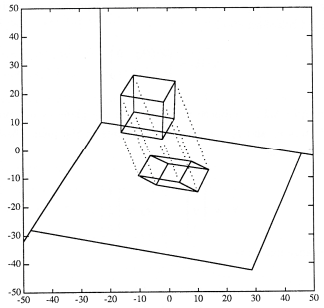
- Step 4: Point clouds from subsequent frames are aligned using Iterative Closest Point (ICP)
  - ICP returns a 6dof orientation matrix
  - Uses the multi-resolution pyramid for performance



REWIRE

Kinect Fusion: Algorithm

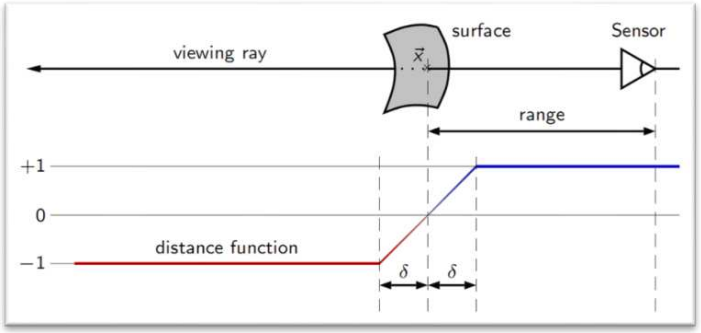
- ICP is modified with a few optimizations:
  - Assume small changes in the camera movement
  - the two clouds are projected onto the same image
  - Points are matched if they fall on the same pixel



REWIRE

Kinect Fusion: Algorithm

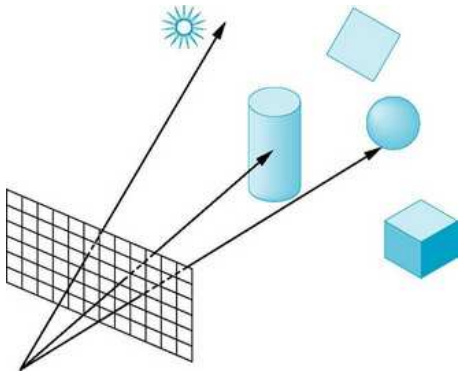
- Step 5: The new cloud is merged with the current compound model to create the surface
  - The surface is extracted using a Truncated Signed Distance Function (TSDF)



REWIRE

Kinect Fusion: Algorithm

- Step 6: Raycasting is performed to render the surface



REWIRE

Kinect Fusion: Algorithm

- Final result!

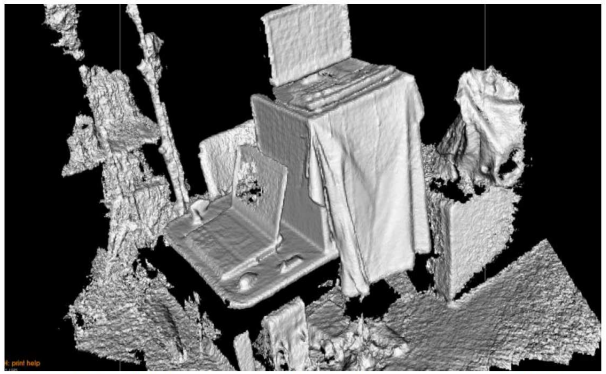


Figure 2 - Part of the AIS-lab at the University of Milan, reconstructed with Kinect Fusion

REWIRE

Thanks for your attention!

Michele Pirovano: michele.pirovano@unimi.it