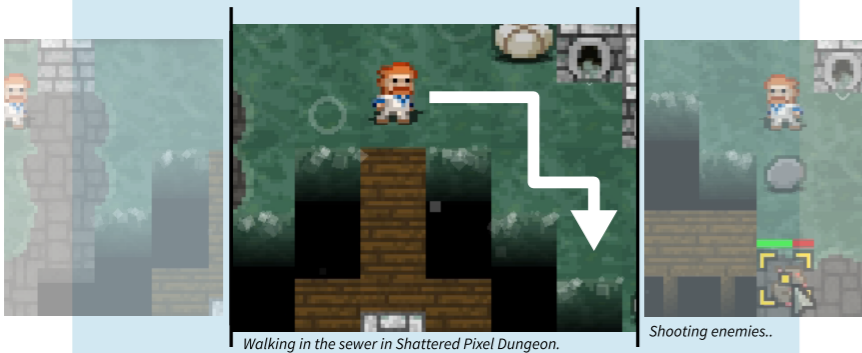


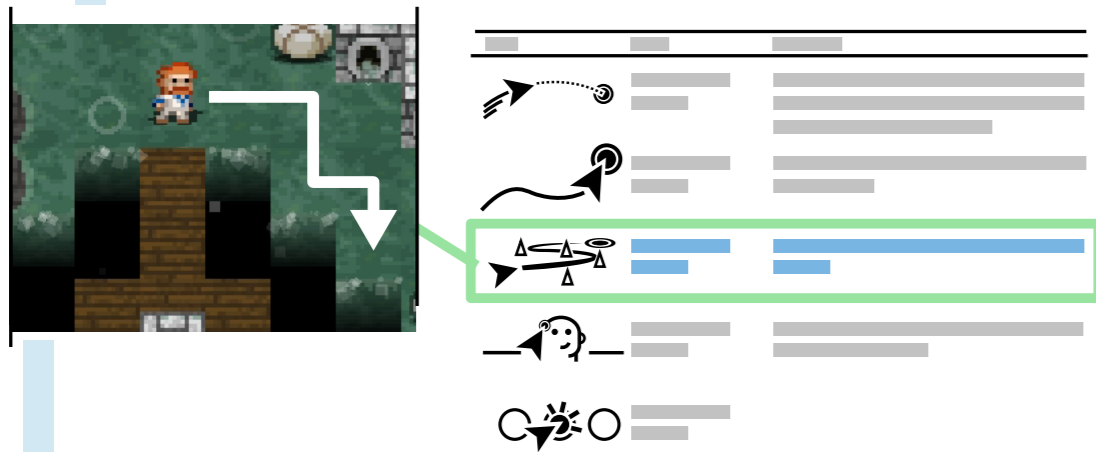
# How to Identify a Core Task and Core Task Properties

## STEP-BY-STEP



**1** Choose a segment from a game you wish to analyse. We suggest making a video recording of the game and analysing it.

**3** Compare the game segment to the core task list. Identify one or more candidates you think could match the interactions in the segment.

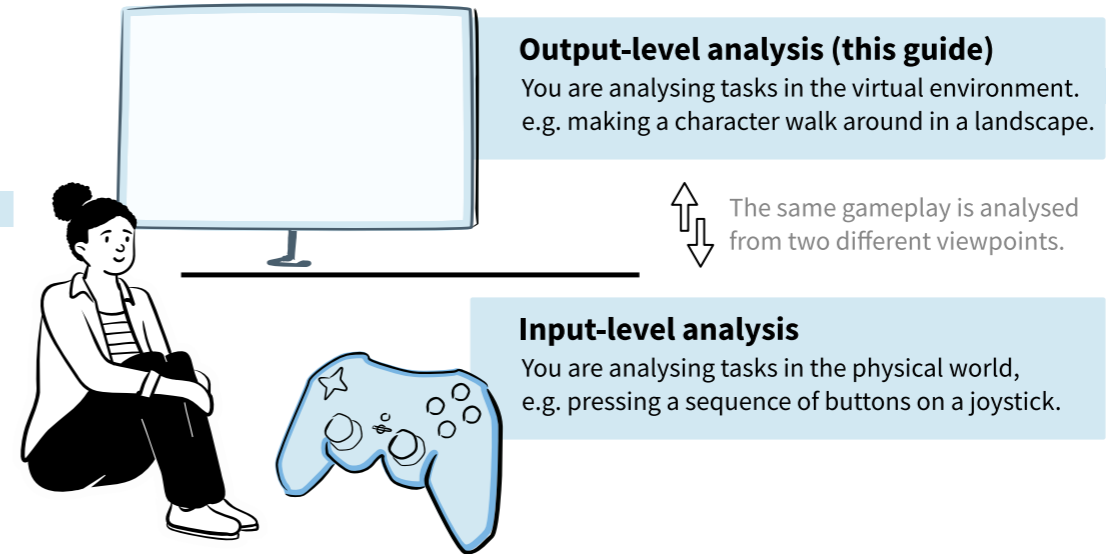


**4** Verify which core task matches your game segment, by closely inspecting its definition and by reading the task criteria listed in the supplementary material for the core task.

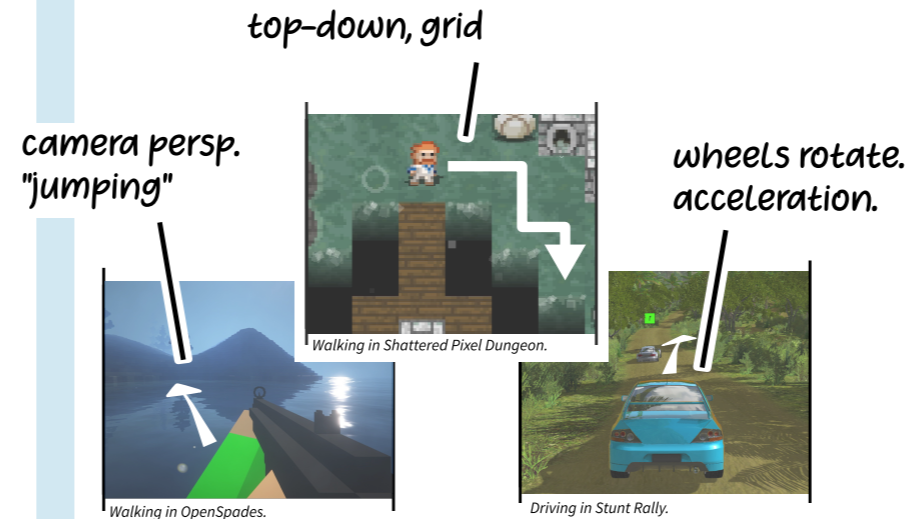
**i** The gameplay may correspond to multiple tasks happening in parallel. In that case, analyse each case separately from each other, before analysing them together.

STEERING TASK	
	<b>Definition:</b> _____
<b>Sub-concepts:</b>	<b>Criteria:</b>
1	1 _____
2	2 _____
3	3 _____
4	4 _____

**2** Determine what kind of analysis you are trying to make? Note that in this guide we will focus on output-level analysis.



**5** To identify task properties, take a look at similar games in this task category. Observe how they differ:  
 → What elements of the environment change the tasks nature?  
 → Are the task sub-levels different?  
 e.g. in what way are actions taken in other order?



**6** Find sensible ways to abstract away from the details specific to each gameplay, that can encompass the different dimensions the games differ. Think of a characteristic in one game and ask yourself "what does this mean in the other game? what changed?"

**degrees of freedom**  
The number of dimensions in which movement is possible.

**movable element**  
What element(s) are being moved.

**spatial information**  
What spatial information is provided for players to orient themselves.

**movement unit**  
What constitutes a minimum amount of movement.

**grid**  
#



**7** Think about the implications of each property you found.

Think of abstract labels categorizing how the property can vary.

**freeform**  
"Freeform steering is in principle possible in any direction at all times."

**track**  
"track-based steering implies that steering must follow a specified track, giving the game designer higher control of e.g. steering difficulty."

How does choosing one or another affect e.g. difficulty, pace, or other experiential qualities?

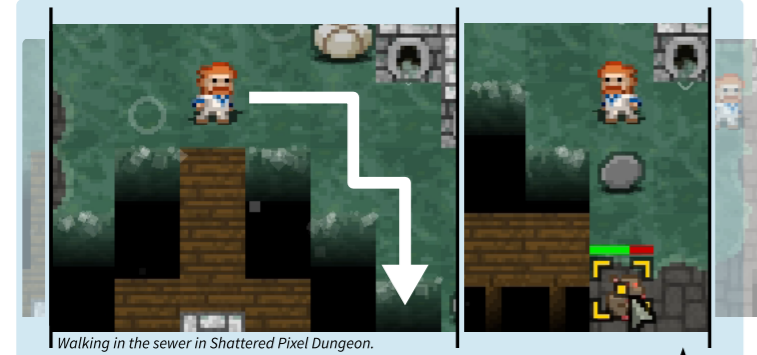
What are the consequences for the player or game environment?

**! Abstraction Pitfalls**  
Getting the concept right is the most difficult. Pay extra attention to the implication of your wording. Always verify how the concept is labeled in relevant scientific literature, if possible.

- 1 vehicle type**  
→ Property not abstract enough. (e.g. wrongly assumes task always uses vehicle)
- 2 car type**  
→ Properties should not overlap one another. (car type overlaps with vehicle type)
- 3 type**  
→ Property label too abstract (type is ambiguous).
- 4 VR-compatible**  
→ Properties cannot imply a specific input device in output-level analysis.
- 5 2D/3D**  
→ The property doesn't necessarily affect the task, may not be unique to the task and may be too imprecisely worded.

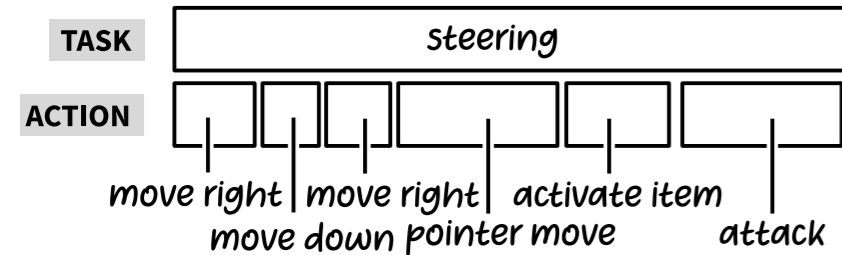
# How to Break Down Tasks and Identify Feedback & Feedforward

## STEP-BY-STEP

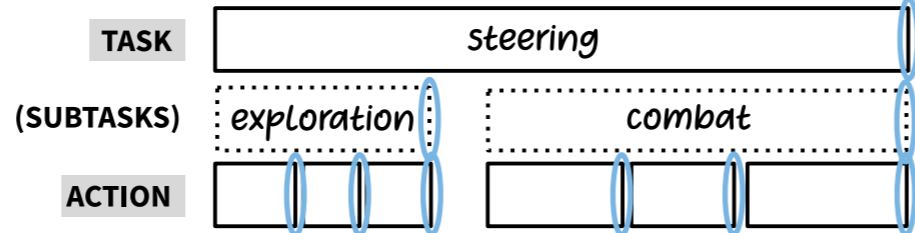
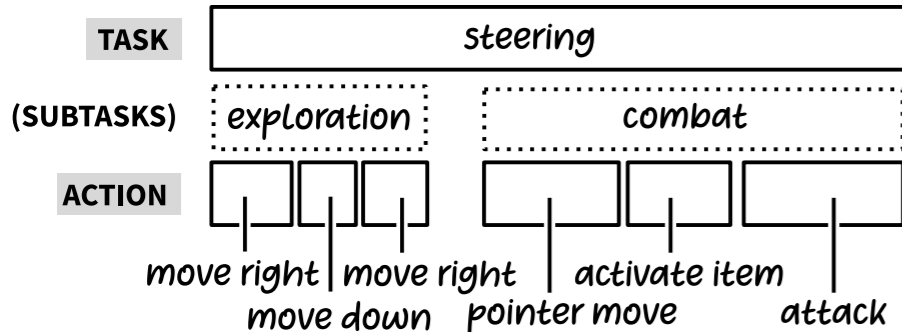


**1** Identify the concrete scenario you are analysing. What kind of task is it? (see also *How to Identify a Core Task and Core Task Properties* guide)

**2** Break the task down into the actions users are required to take in order to solve it, by chronologically inspecting every user's interaction from e.g. a game video.



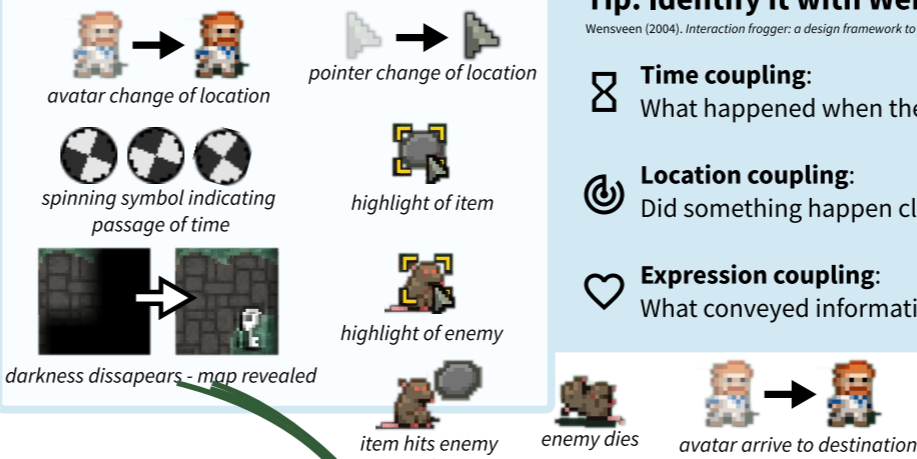
**3** Optionally, if your task covers many actions, check whether it makes sense to further group your actions into subtasks.



**4** We now consider feedback opportunities in our model. Feedback can typically happen at the end of any action or task block (highlighted above with blue).

**5** Next, we list what feedback can be observed from the game's gameplay.

### Feedback List

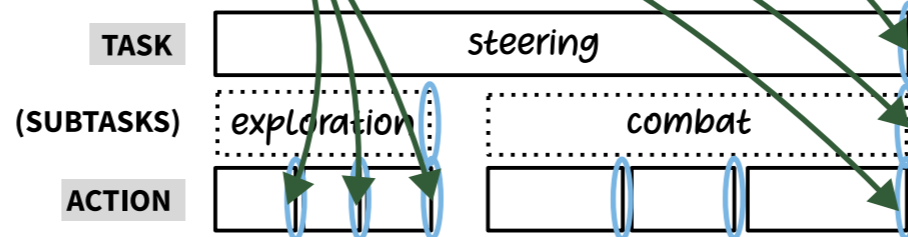


### Tip: Identify it with Wensveen's Natural Couplings:

Wensveen (2004). *Interaction frogger: a design framework to couple action and function through feedback and feedforward*

- Time coupling:** What happened when the user performed the action/task?
- Location coupling:** Did something happen close to any user-controlled entity?
- Expression coupling:** What conveyed information related to the users action/task?

**6** Now link the identified feedback in the game's gameplay to the game's task model.



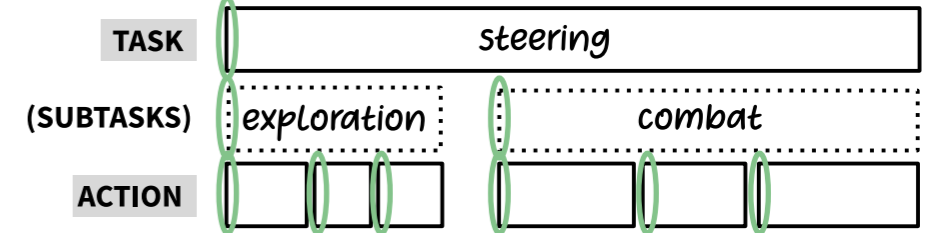
**1 Where To Link?** To match feedback in time to your model, ask yourself after what action does the feedback occur and is the feedback a consequence of that action?

**2 What level?** To match the feedback to a level (action/subtask/task), ask yourself what information is the feedback providing?

### Feedback Rules of Thumb:

At *action-level*, feedback typically tells users whether the single action was successful and potentially how or why.

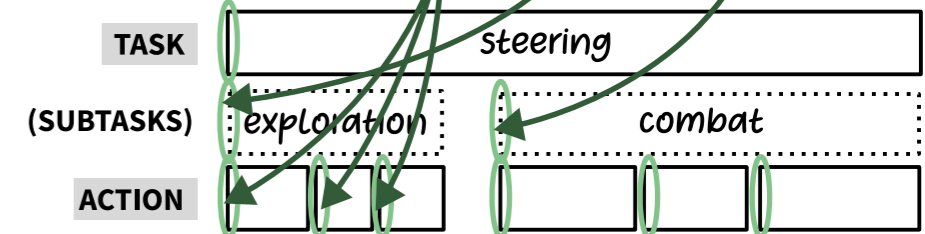
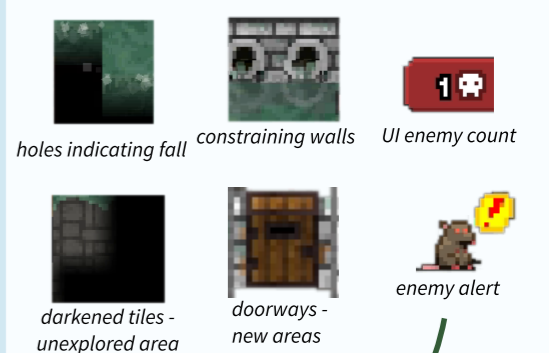
At *task-level*, feedback typically tells the result of the combined actions.



**7** Feedforward typically takes place *before* an action or task, which we now focus on (highlighted green).

**8** Next, list what feedforward you observe from the game's gameplay and link them to your model.

### Feedforward List



### Feedforward Rules of Thumb:

At *action-level*, feedforward provides information related to the outcomes of an action.

At *task-level*, feedforward provides information in advance related to the possible outcomes of a task.

**9** What you have in your resulting model, is an overview of feedback and feedforward employed by the game and identified opportunities for more of it. Now, analyse:



### Opportunities for Analysis:

→ Any place *with* feedback/forward shows a design decision. Think about what could be alternative types of feedback/forward in these places?

→ Any place *without* feedback/forward marks opportunities for feedback/forward. how would feedback/forward have looked like, if it was present in these places?