Exercise
Advanced Programming Techniques
Project Phase

Martin Bauer, Sebastian Kuckuck

Chair for System Simulation
Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Project Schedule

- **Phase 1**: planning phase
  - team formation
  - agree on software infrastructure (build system, version control, …)
  - software design: UML diagrams
- **Phase 2**: forward simulation

- ***Phase 3****: optimization (starting after Christmas break)*

Team formation

- form teams of size 3
- use forum to find a group or additional group members
- each group member is responsible to implement one of the three SC2 races – bonus points in exam
- register your group on StudOn (please stick to format!)
- form group **until next week Wednesday**

```
Please add your team formations here (Team name, member names and race assignments)

- TeamAwesome
   - Martin Bauer <Zerg>
   - Sebastian Kuckuk <Terran>
   - Markus Obereisenbuchner <Protoss>
```

Software Infrastructure

Version Control System

- use version control system to collaborate in group
- preferred choice: git
- Can be hosted here: https://gitlab.cs.fau.de
  - no public github repositores!

Build System

- agree on build system (preferably CMake)

Star Craft Modelling: Resources

Minerals

- can be harvested directly
- collection rate per worker per second

Vespene Gas

- only 2 gas geysers per base
- to collect gas, a building has to be constructed on geyser (geyser has to be “tapped”)
- maximum 3 workers per tapped geyser

Star Craft Modelling: Resources

Worker Distribution

- worker distribution can be chosen by forward simulator
- possible strategies: half/half, all possible to gas - rest to minerals, dependent on build-list

Supply (“Housing”)

- units require supply
- supply is generated by certain buildings
- supply has to be available when unit build process is started

Star Craft Modelling: Construction

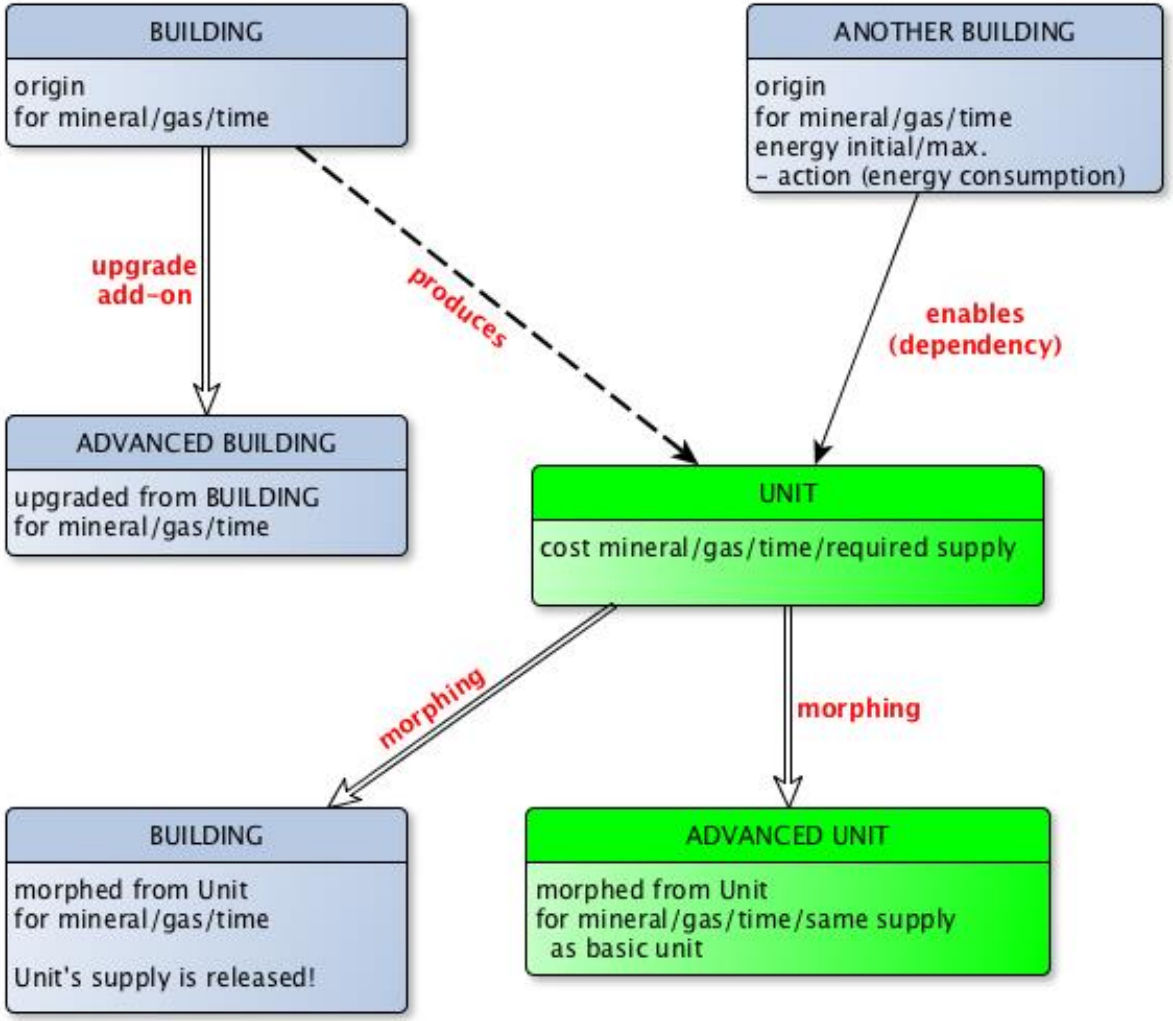
Races

- three races: *Terran, Zerg, Protoss*
- each race has different buildings, units & special abilities
- also the building and unit creation process is different

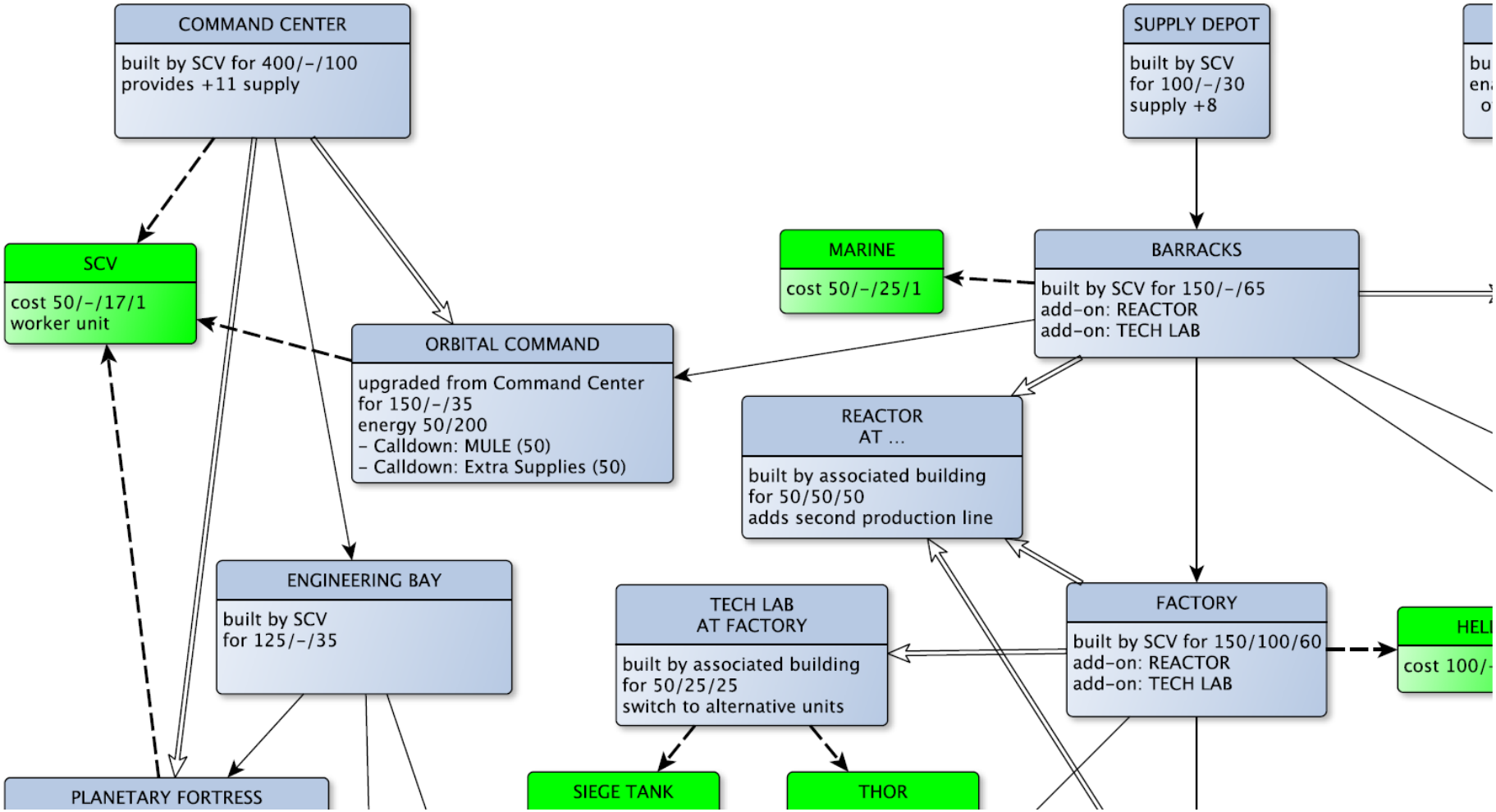
Common Properties

- resources are deducted when build process starts
- buildings / units have dependencies i.e. some buildings or units can only be built if a certain other building already exists
- these dependencies are specified in a “tech-tree”

Star Craft Modelling: Tech Tree



Star Craft Modelling: Terran



Star Craft Modelling: Terran

Construction

- worker (SCV) is occupied while creating a building
- units are created in buildings which are occupied during unit creation
- supply is provided by building *supply depot*

Special Abilities

- upgraded main building has an *energy* associated with it, which loads up with a constant rate and can be used to triggered special abilities
- *MULE:* temporary unit (90 s), which is the equivalent of 4 workers and can be used to collect minerals. Does not cost minerals, gas or supply
- *Extra Supplies:* can be applied to supply depot, to give permanent, one-time increase of supplies – usually not worth to implement it, since energy is better used for MULEs

Star Craft Modelling: Zerg

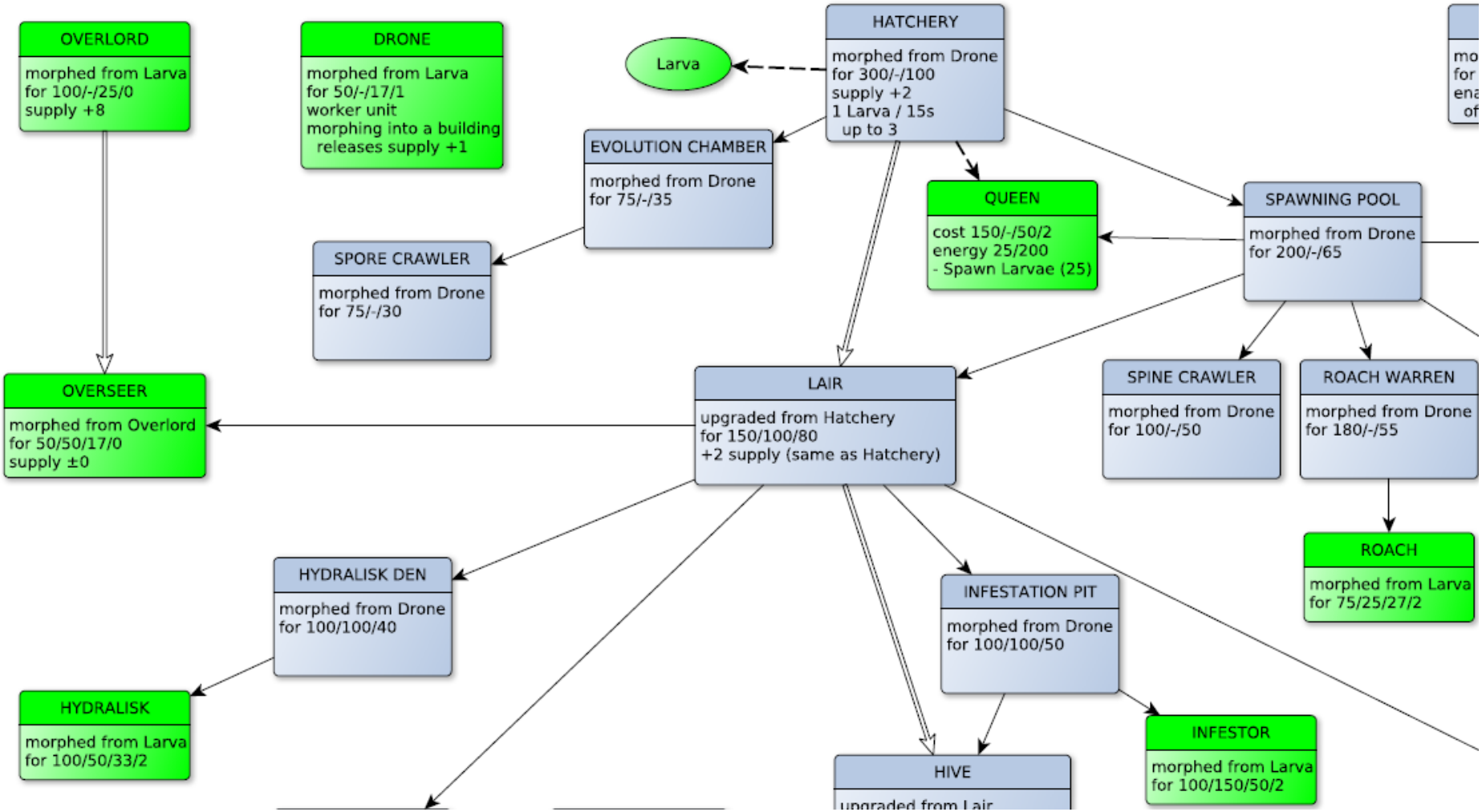
Construction

- buildings are created by “morphing” i.e. the creating worker unit (drone) is deleted when build process starts
- units are not built, but morphed from other units
- starting unit “larva”, are spawning regularly (one every 15s) at main building up to a limit of 3

Special Abilities

- special unit: queen has associated energy, loading up at constant rate. Each queen has separate energy value
- *Inject larva:* place 4 eggs into main building, which become larvae after 40 s

Star Craft Modelling: Zerg



Star Craft Modelling: Protoss

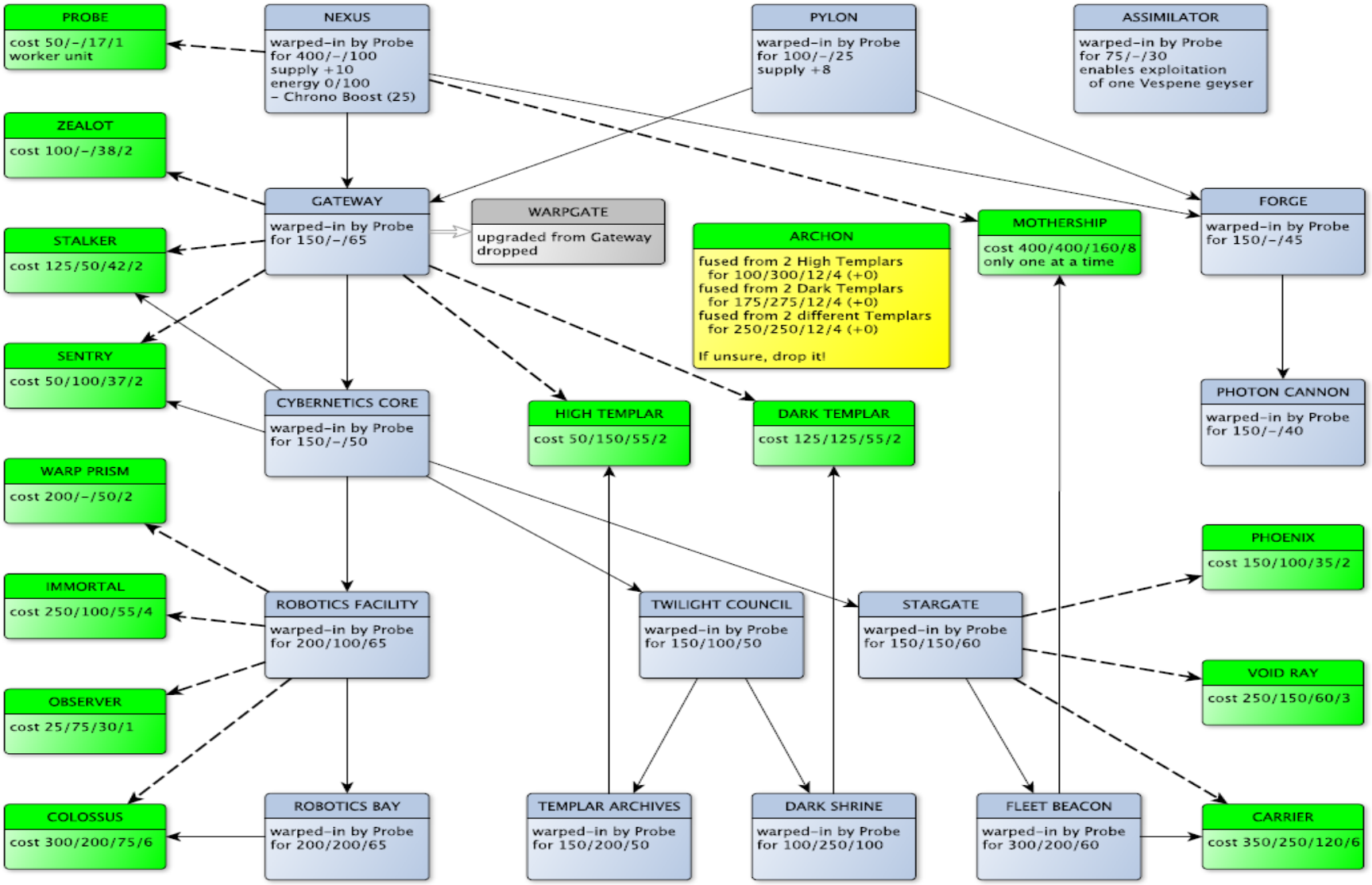
Construction

- worker is not occupied by construction – i.e. for our purposes not even needed

Special Abilities

- energy associated with main building
- *Chrono Boost:* speed up production rate of one building by 50% for 20s

Star Craft Modelling: Protoss



Tech Trees

- some data also available in machine readable form (CSV)
- optional: you can use the file as it is, modify it, or not use at all

#name	minerals	vespene	build time	supply cost	supply provided	start energy	max energy	race	produced_by	dependency
probe	50	0	17	1	0	0	0	protoss	nexus	
zealot	100	0	38	2	0	0	0	protoss	gateway	
stalker	125	50	42	2	0	0	0	protoss	gateway	cybernetics_core
sentry	50	100	37	2	0	0	0	protoss	gateway	cybernetics_core
warp_prism	200	0	50	2	0	0	0	protoss	robotics_facility	
immortal	250	100	55	3	0	0	0	protoss	robotics_facility	
observer	25	75	30	1	0	0	0	protoss	robotics_facility	
colossus	300	200	75	6	0	0	0	protoss	robotics_facility	robotics_bay
high_templar	50	150	55	2	0	0	0	protoss	gateway	templar_archives

Project Phase 1

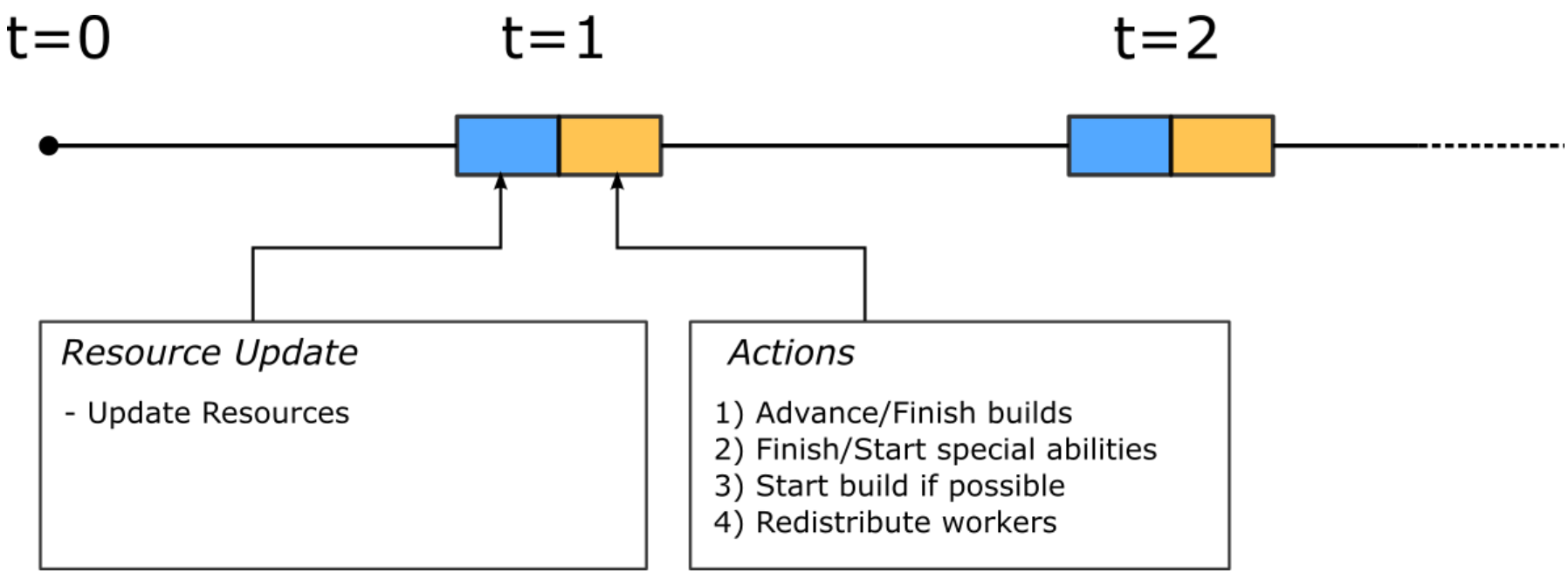
- team formation, repository setup
- plan your software
  - define requirements
  - create sub-tasks and assign to team members
  - plan your implementation e.g. UML class diagrams
- next week: show your plans to AdvPt tutors in computer exercise
- use this opportunity for early feedback: bad design can lead to lots of work later on

Project Phase 2: Forward Simulation

- Input: Build lists
  - text file with one unit per line
  - has to be built **in this order**
- first, decide if build list is valid
- if it is valid:
  - write detailed log in JSON format
  - you can use library for JSON output as long as it is build along with your project
- Examples and validation at
  - **https://www10.cs.fau.de/advptSC2/**

```
scv
scv
scv
supply_depot
scv
barrack
marine
```

Project Phase 2: Forward Simulation



Project Phase 2: Forward Simulation


```
{
    "time": 1,
    "status": {
      "workers": {
        "minerals": 6,
        "vespene": 0
      },
      "resources": {
        "minerals": 4,
        "vespene": 0,
        "supply-used": 7,
        "supply": 11
      }
    },
    "events": [
      {
        "type": "build-start",
        "name": "scv"
      }
    ]
  },
```

Outlook: Phase 3 - Optimization

- starts after Christmas break
- write a build list optimizer using a genetic algorithm (or branch & bound)
- **beneficial if forward simulation is fast**
- two optimization scenarios:
  - build as many units as possible in given time
  - build a certain (advanced) unit / building as fast as possible

- contest between groups

Project

Questions?