Design Document

Project XScorch
Team Core Dump
February 19, 2003

Change Record:

Date	Version	Changes/Additions	Responsible Person
2/7/03	1.0	Created document	Jamil Daouk, Jeremy Stolarz, & Iggy Villavelazquez
2/19/03	1.5	Added UML structure diagrams for classes other than sc_player, fixed other issues with document as noted by grader's comments	Alon Gotesman, Iggy Villavelazquez, Jamil Daouk, & Jeremy Stolarz

Table of Contents

Overview

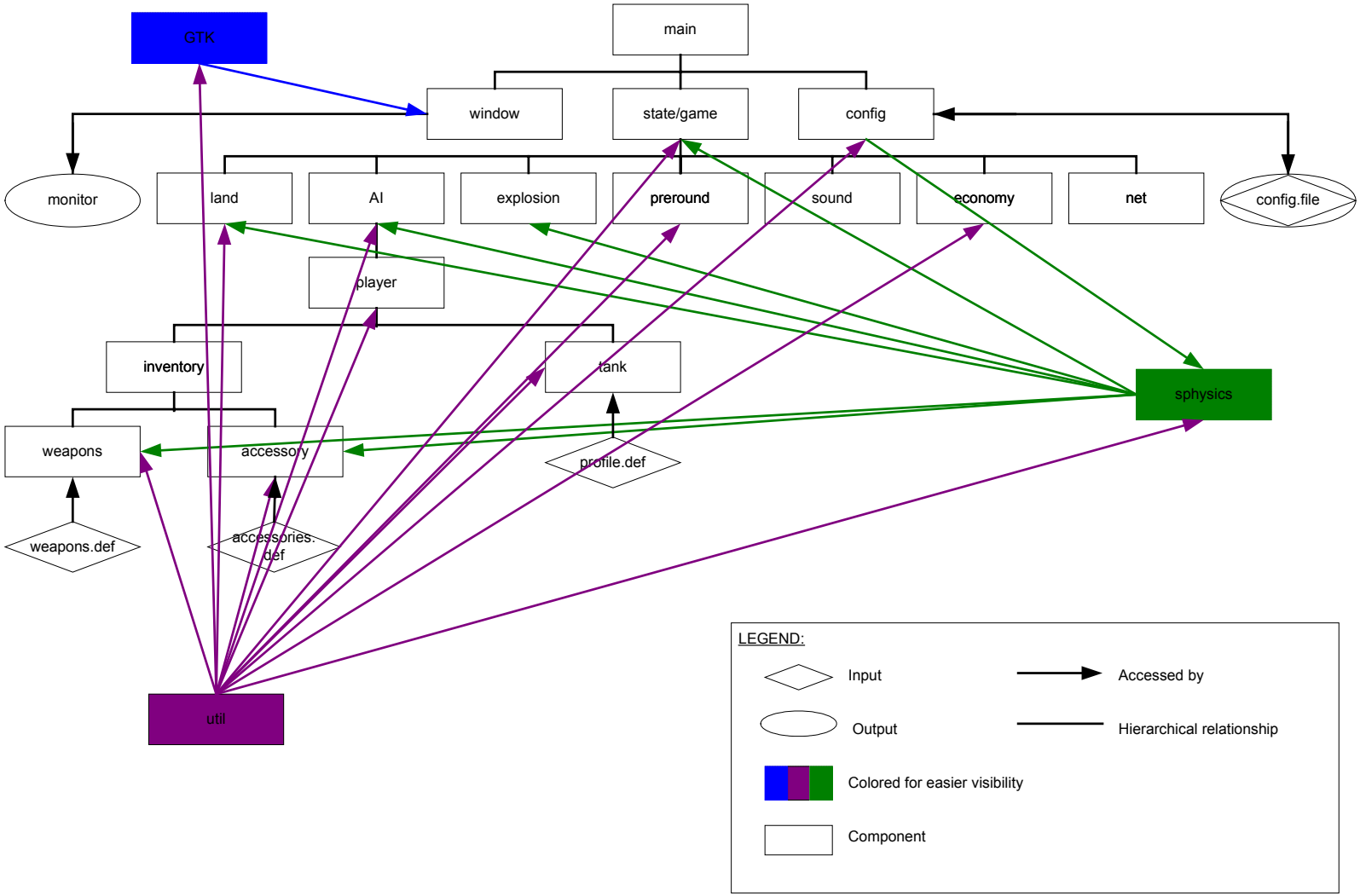
This document shows the design of Xscorch, focusing on areas that we are going to change. The architecture diagram shows the entire project conceptually, in a simple hierarchical form. The UML structure diagrams show the details of the significant classes for the project and their relationships. The sequence diagrams trace the route of execution for scenarios that are integral to the implementation of our new features.

Major Design Decisions

Most of our features do not require the creation of new classes or objects. Rather, they merely require changes in current classes like new data and new functions. We deem that one of the biggest challenges will be creating the lightning and meteors for the hostile environment. We plan to make a new weapon type for these natural hazards and use existing weapon tracking functions to resolve them. Another issue that concerns us with this feature is its implementation in synchronous mode. We will probably have it occur once per set of turns.

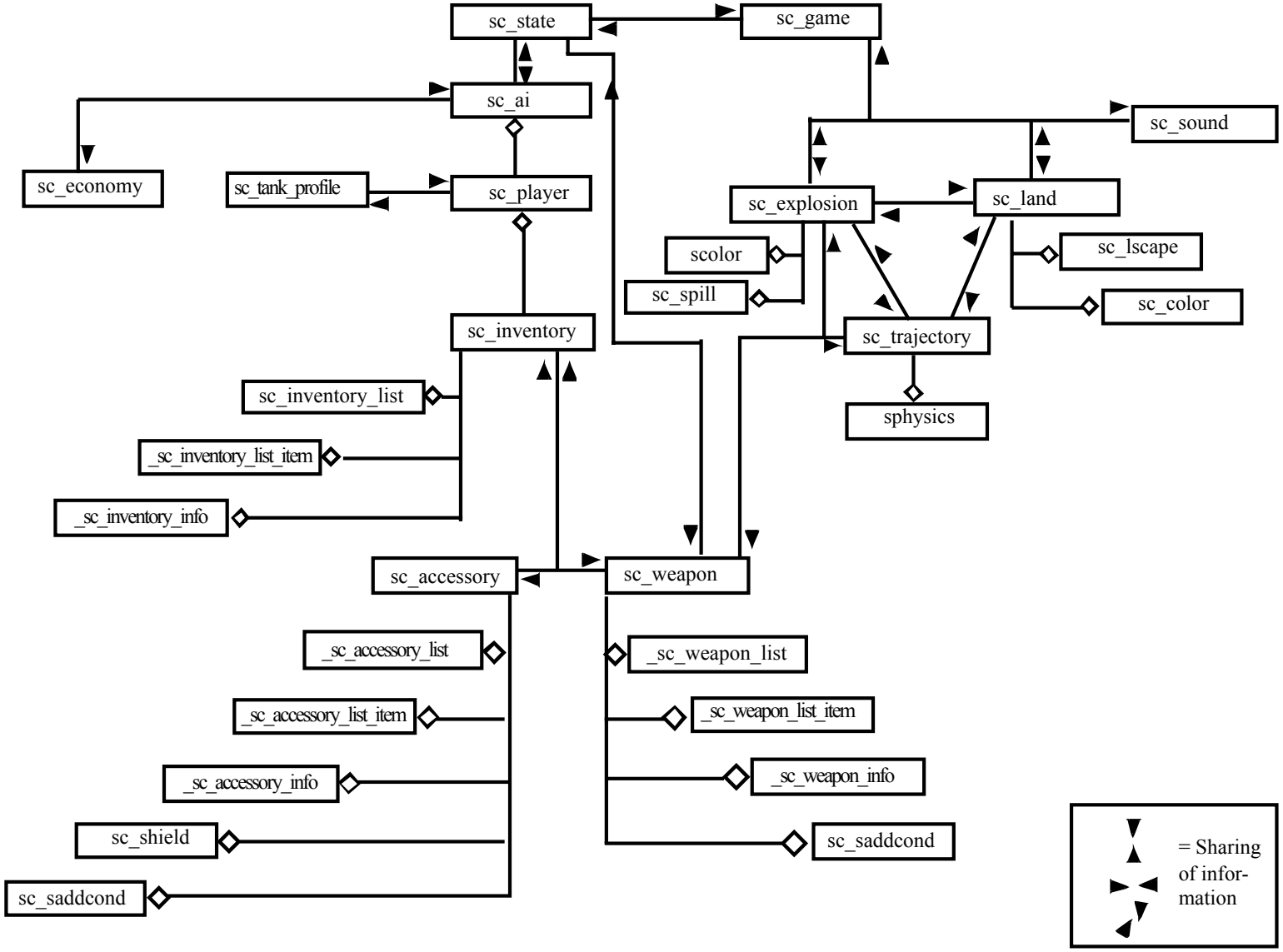
Architecture

The architecture diagram on the following page is a conceptualization of the overall program. The components are not necessarily classes. They can have multiple classes within them. The util, sphysics, and GTK components do not fit into the hierarchical structure, rather they supply the other components with necessary functions. An arrow goes from config to sphysics because sphysics uses the current configuration to make some of its computations. Components in a hierarchical relationship mostly communicate in two directions and are fairly intertwined. They do not have arrows between them for greater simplicity of the diagram.


GTK
main
window
state/game
config
monitor
land
AI
explosion
preround
sound
economy
net
config.file
player
inventory
tank
sphysics
weapons
accessory
profile.def
weapons.def
accessories.def
util
LEGEND:
Input
Accessed by
Output
Hierarchical relationship
Colored for easier visibility
Component

Structure Diagrams

The first structure diagram shows the relationships between all the important classes in the program. However, it does not show the composition of the classes for space issues the see the subsequent diagrams for detailed descriptions of the major classes (those that are going to require the most modifications by the implementation of our new features).


sc_state
sc_game
sc_ai
sc_sound
sc_economy
sc_tank_profile
sc_player
sc_explosion
sc_land
scolor
sc_lscape
sc_spill
sc_color
sc_inventory
sc_trajectory
sc_inventory_list
sphysics
_sc_inventory_list_item
_sc_inventory_info
sc_accessory
sc_weapon
_sc_accessory_list
_sc_weapon_list
_sc_accessory_list_item
_sc_weapon_list_item
_sc_accessory_info
_sc_weapon_info
sc_shield
sc_saddcond
sc_saddcond
= Sharing of infor-mation

sc_player
+ index : int + name[SC_PLAYER_NAME_LENGTH] : char + turret : int + power : int + turret : int + power : int + life : int + dead : bool + fuel : int + x : int + y : int + field_index : int + numwins : int + kills : int + suicides : int + killedby : int + money : int + oldmoney : int + armed : bool + armslevel : int + contacttriggers : bool + ac_state : int - _sc_player_messages[SC_PLAYER_NUM_MESSAGES + 1] : char* - _sc_player_death_messages[SC_PLAYER_NUM_DEATH_MESSAGES + 1] : char*
+ sc_player_new(index : int, tank : _sc_tank_profile*) + sc_player_free(p : sc_player**) + sc_player_init_game(c : _sc_config*, p : sc_player*) + sc_player_init_turn(c : _sc_config*, p : sc_player*) + sc_player_advance_power(c : _sc_config*, p : sc_player*, delta: int) + sc_player_advance_turret(c : _sc_config*, p : sc_player*, delta: int) + sc_player_advance_weapon(c : _sc_config*, p : sc_player*, delta: int) + sc_player_advance_shield(c : _sc_config*, p : sc_player*, flags: int) + sc_player_set_power(c : _sc_config*, p : sc_player*, power : int) + sc_player_set_turret(c : _sc_config*, p : sc_player*, turret : int) + sc_player_set_weapon(c : _sc_config*, p : sc_player*, info : _sc_weapon_info) + sc_player_set_shield(c : _sc_config*, p : sc_player*, info : _sc_accessory_info) + sc_player_activate_shield(c : _sc_config*, p : sc_player*) + sc_player_activate_best_shield(c : _sc_config*, p : sc_player*) + sc_player_activate_battery(c : _sc_config*, p : sc_player*) + sc_player_activate_teleport(c : sc_config*, p : sc_player*) + sc_player_set_position(c : sc_config*, p : sc_player*, x : int, y : int) + sc_player_set_contact_triggers(c : _sc_config*, p : sc_player*, flag : bool) + sc_player_use_contact_trigger(c : sc_config*, p : sc_player*) + sc_player_battery_count(c : sc_config*, p : sc_player*) + sc_player_contact_trigger_count(c : sc_config*, p : sc_player*) + sc_player_inc_wins(c : sc_config*, p : sc_player*) + sc_player_died(c : sc_config*, p : sc_player*) + sc_player_drop_all(c : sc_config*) + sc_player_damage_all(c : sc_config*, e : _sc_explosion*) + sc_player_death(c : sc_config*, p : sc_player*, e : _sc_explosion*)

```
+ sc_player_talk(c : sc_config*, p : sc_player*)
+ sc_player_death_talk(c : sc_config*, p : sc_player*)
+ sc_player_random_order(c : sc_config*, playerlist : sc_player**)
+ sc_player_winner_order(c : sc_config*, playerlist : sc_player**)
+ sc_player_loser_order(c : sc_config*, playerlist : sc_player**)
+ sc_player_total_fuel(ac : _sc_accessory_config*, p : sc_player*)
+ sc_player_move(c : sc_config*, p : sc_player*, delta : int)
+ sc_player_passable(c : sc_config*, p : sc_player*, x : int, y : int)
+ sc_player_turret_x(p : sc_player*, angle : int)
+ sc_player_turret_y(p : sc_player*, angle : int)
+ sc_player_would_impact(c : sc_config*, p : sc_player*, x : int, y :
    int)
+ sc_player_get_inventory_value(c : sc_config*, p : sc_player*)
- _sc_player_drop(c : sc_config*, p : sc_player*, falldist : int)
- _sc_player_damage(c : sc_config*, p : sc_player*, e : _sc_explosion*)
- _sc_player_consume_fuel(ac : _sc_accessory_config*, p : sc_player*)
```

sc_inventory

```
- _sc_inventory_quantity(current : int , bundlesize: int)
- _sc_inventory_purchase_price(info : sc_inventory_info*, quantity :
    int)
- _sc_inventory_sale_price(info: sc_inventory_info*, quantity: int)
+ sc_inventory_can_buy_weapon(p: sc_player*, info : sc_weapon_info*,
    budget: int)
+ sc_inventory_buy_weapon(p: sc_player*, info : sc_weapon_info*)
+ sc_inventory_can_buy_accessory(p : sc_player*, info:
    sc_accessory_info*, budget: int)
+ sc_inventory_buy_accessory(p: sc_player*, info : sc_accessory_info*)
+ sc_inventory_can_sell_weapon(p: sc_player*, info: sc_weapon_info*)
+ sc_inventory_sell_weapon(p: sc_player*, info: sc_weapon_info*)
+ sc_inventory_can_sell_accessory(p: sc_player*, info:
    sc_accessory_info*)
+ sc_inventory_sell_accessory(p: sc_player*, info: sc_accessory_info*)
+ sc_inventory_award_weapon(info: sc_weapon_info*, player: int)
```

sc_inventory_list

```
+ defs_hash[SC_INVENTORY_HASH_SIZE]: sc_inventory_list_item*
+ defs_head: sc_inventory_list_item*
+ defs_tail: sc_inventory_list_item*
```

_sc_inventory_list_item

```
+ prev: _sc_inventory_list_item*
+ next: _sc_inventory_list_item*
+ chain: _sc_inventory_list_item*
+ item: _sc_inventory_info*
```

_sc_inventory_info

```
+ ident: int
+ armslevel: int
+ price: int
+ bundle: int
```

sc_economy
+ interestrate : double + dynamicinterest : bool + currentinterest : double + initialcash : int + computersbuy : bool + computersaggressive: bool + freemarket : bool + lottery : bool + scoring : sc_economy_scoring + survivalbonus : int + damagebonus : int + killbonus : int + damageloss : int + deathloss : int + suicideloss : int
+ sc_economy_new() + sc_economy_free(ec : sc_economy**) + sc_economy_init(ec : sc_economy*) + sc_economy_interest(c: _sc_config*, ec : sc_economy*) + sc_economy_scoring_names() + sc_economy_scoring_types()

sc_accessory
+ sc_accessory_count(ac : sc_accessory_config*, flags : int) + sc_accessory_first(ac : sc_accessory_config*, flags : int) + sc_accessory_last(ac : sc_accessory_config*, flags : int) + sc_accessory_lookup(ac : sc_accessory_config*, id : int, flags : int) + sc_accessory_lookup_by_name(ac : sc_accessory_config*, name : char*, flags) + sc_accessory_prev(ac : sc_accessory_config*, sc_accessory_info* : info, flags : int) + sc_accessory_next(ac : sc_accessory_config*, sc_accessory_info* : info, flags : int) + sc_accessory_config_free(ac : sc_accessory_config**) + sc_accessory_inventory_clear(ac: sc_accessory_config*) + sc_accessory_config_new()

sc_accessory_list
+ defs_hash[SC_ACCESSORY_HASH_SIZE]: sc_accessory_list_item* + defs_head: sc_accessory_list_item* + defs_tail: sc_accessory_list_item*
+ sc_accessory_list_new() + sc_accessory_list_free(al: sc_accessory_list**)

sc_accessory_list_item
+ prev: _sc_accessory_list_item* + next: _sc_accessory_list_item* + chain: _sc_accessory_list_item* + item: _sc_accessory_info*
+ sc_accessory_list_item_new() + sc_accessory_list_item_free(ali: sc_accessory_list_item**)

sc_accessory_info
+ ident: int + armslevel: int + price: int + bundle: int + shield: int + fuel: int + repulsion: int + state: int + inventories[SC_MAX_PLAYERS]: int + useless: bool + indirect: bool + name: char* + description: char*
+ sc_accessory_info_line(ac : sc_accessory_config*, sc_accessory_info* : info, buffer : char*, buflen : int) + sc_accessory_info_free(ai: sc_accessory_info**)

sc_ai
+ dx: int + dy: int + index: int + counter: int + outer: int + inner: int + percent: double + result: bool - humantargets: bool - allowoffsets: bool - alwaysoffset: bool - enablescan : bool - nobudget: bool
+ sc_ai *sc_ai_new( ); + sc_ai_free(ai: sc_ai**); + sc_ai_controller *sc_ai_controller_new( ); + sc_ai_controller_free(aic: sc_ai_controller**); + sc_ai_init_game(c: _sc_config*, p: _sc_player *p); + sc_ai_init_round(c: _sc_config*, p: _sc_player*); + sc_ai_player_turn(c: _sc_config*, p: _sc_player*); + sc_ai_player_buy(c: _sc_config*, p: _sc_player*); + sc_ai_trajectory_terminus(c: _sc_config*, tr: _sc_trajectory*); + sc_ai_trajectory(c: _sc_config*, p:_sc_player*,victim: _sc_player*); + sc_ai_trajectory_line(c: _sc_config*, p: _sc_player*, victim: _sc_player*); + sc_ai_trajectory_wind(c: _sc_config*, p: _sc_player*,victim _sc_player*); + sc_ai_trajectory_line_wind(c: _sc_config*, p: _sc_player*, victim: _sc_player*); + sc_ai_trajectory_scan(c: _sc_config *c, p: _sc_player*, victim: _sc_player*); - _sc_ai_trajectory_compensation(c: sc_config*,p: sc_player*, victim: sc_player*, deltax: int*, deltay: int*) - _sc_ai_trajectory_angle_bounds(deltax: int,deltay: int, maxdelta:int) - _sc_ai_validate_angle(c:sc_config*,p: sc_player*, angle: int) - _sc_ai_trajectory_scan(c: sc_config*, p: sc_player*, victim: sc player*, newpower: int*, newangle:int*, minangle:int, maxangle:int,

```
deltaangle: int, minpower:int, maxpower: int, deltapower: int,
maxdist:double)
- _sc_ai_select_last_weapon(c:sc_config*,p:sc_player*)
- _sc_ai_select_weapon_by_score(c:sc_config*,p: sc_player*)
- _sc_ai_select_shield_sappers(c:sc_config*,p: sc_player*, target:
sc_player*)
- _sc_ai_random_fire(c: sc_config*,p: sc_player*)
- _sc_ai_line_of_sight(c:sc_config*,p: sc_player*,vp:sc_player*, line:
trajfn,noline:trajfn)
- _sc_ai_target_practice(c:sc_config*,playerlist:sc_player**)
- _sc_ai_calculated_fire_wind(c: sc_config*, p:sc_player*)
- _sc_ai_fire_at_victim(c:sc_config*, p: sc_player*)
- _sc_ai_fire_at_victim_ruthlessly(c: sc_config*,p: sc_player*)
- _sc_ai_raise_shields(c: sc_config*,p: sc_player*)
- _sc_ai_set_contact_triggers(c: sc_config*,p: sc_player*)
- _sc_ai_recharge_tank(c: sc_config*, p: sc_player*)
- _sc_ai_turn_status(c: sc_config*, p: sc_player*)
```

sc_weapons

```
+ sc_weapon_count(wc: sc_weapon_config*, flags: int);
+sc_weapon_statistic(wc: sc_weapon_config*, info: sc_weapon_info*, p:
_sc_player*, statistic: sc_weapon_stat);
+ sc_weapon_config_free(wc: sc_weapon_config**);
+ sc_weapon_inventory_clear(wc: sc_weapon_config*);
+ sc_weapon_config_new( );
+ sc_weapon_new(c: _sc_config*,i: sc_weapon_info*, x: double, y:
double, vx: double, vy: double,ct: bool,playerid)
+ sc_weapon_fire_new(c: _sc_config*, p: _sc_player*,e: _sc_explosion
**);
+ sc_weapon_landfall(c: _sc_config*, wp: sc_weapon*);
+ sc_weapon_free_chain(wp: sc_weapon**);
+ sc_weapon_free(wp: sc_weapon**);
+ sc_weapon_create_all(c: _sc_config*, e: _sc_explosion**);
+ sc_weapon_print_yields(wc: sc_weapon_config*);
+ sc_weapon_state_bit_names( );
+ sc_weapon_state_bit_items( );
- _sc_weapon_viewable(wc: sc_weapon_config*, info: sc_weapon_info*,
flag: int);
+ sc_weapon_count(wc: sc_weapon_config*, flags: int);
- _sc_weapon_compare_names(A: char*, B: const char*);
- _sc_weapon_stat_yield(wc: sc_weapon_config*, info: sc_weapon_info*)
- _sc_weapon_stat_precision(wc: sc_weapon_config*,
info:sc_weapon_info*, triple: bool);
```

sc_weapons_list

```
+ sc_weapon_list_free(wl: sc_weapon_list**);
+ sc_weapon_list_new( );
```

sc_weapons_list_item

```
+ sc_weapon_list_item_free(li: sc_weapon_list_item**);
+ c_weapon_list_item_new( );
```

sc_weapons_info
+ sc_weapon_first(wc: sc_weapon_config*, flags: int); + sc_weapon_last(wc: sc_weapon_config*, flags: int); + sc_weapon_lookup(wc: sc_weapon_config*,id: int, flags: int); + sc_weapon_lookup_by_name(wc: sc_weapon_config*, name: char*,flags: int); + sc_weapon_prev(wc: sc_weapon_config*, info: sc_weapon_info*,flags: int); + sc_weapon_next(wc: sc_weapon_config*,info: sc_weapon_info*,flags: int); + sc_weapon_info_line(wc: sc_weapon_confi*, info: sc_weapon_info*, buffer: char*, buflen: int); + sc_weapon_info_free(wi: sc_weapon_info**); +_sc_weapon_dump_info(info: sc_weapon_info*);

Sequence Diagrams

1) The sequence diagram shows how music is started for the inventory screen. Starting from the game state machine. "As soon as xscorch starts up Dino's disposition improves dramatically because great melodies begin to emanate from his computer's speakers" (Scenario 3).

2) This diagram shows the sequence of events required for the economist AI to buy its inventory. First it buys contact triggers, then auto-defense, and the best shields available. Afterwards it buys shield zappers and precision weapons. Finally it buys batteries. There is no scenario written for this feature.

3) This sequence diagram shows how the economist AI takes its turn. The different steps it goes through before attacking an opponent. There is no scenario for this feature.

4) This diagram shows the sequence of events for a random lightning strike to occur at the beginning of a players turn. "...Joe notices little meteors shooting across the screen, creating small explosions – much as a normal weapon would! It was a similar case with lightning bolts a few minutes earlier, he recalled" (Scenario 4).

5) This diagram shows the sequence of events for a random meteor strike to occur at the beginning of a players turn. See quote for diagram 4 above.

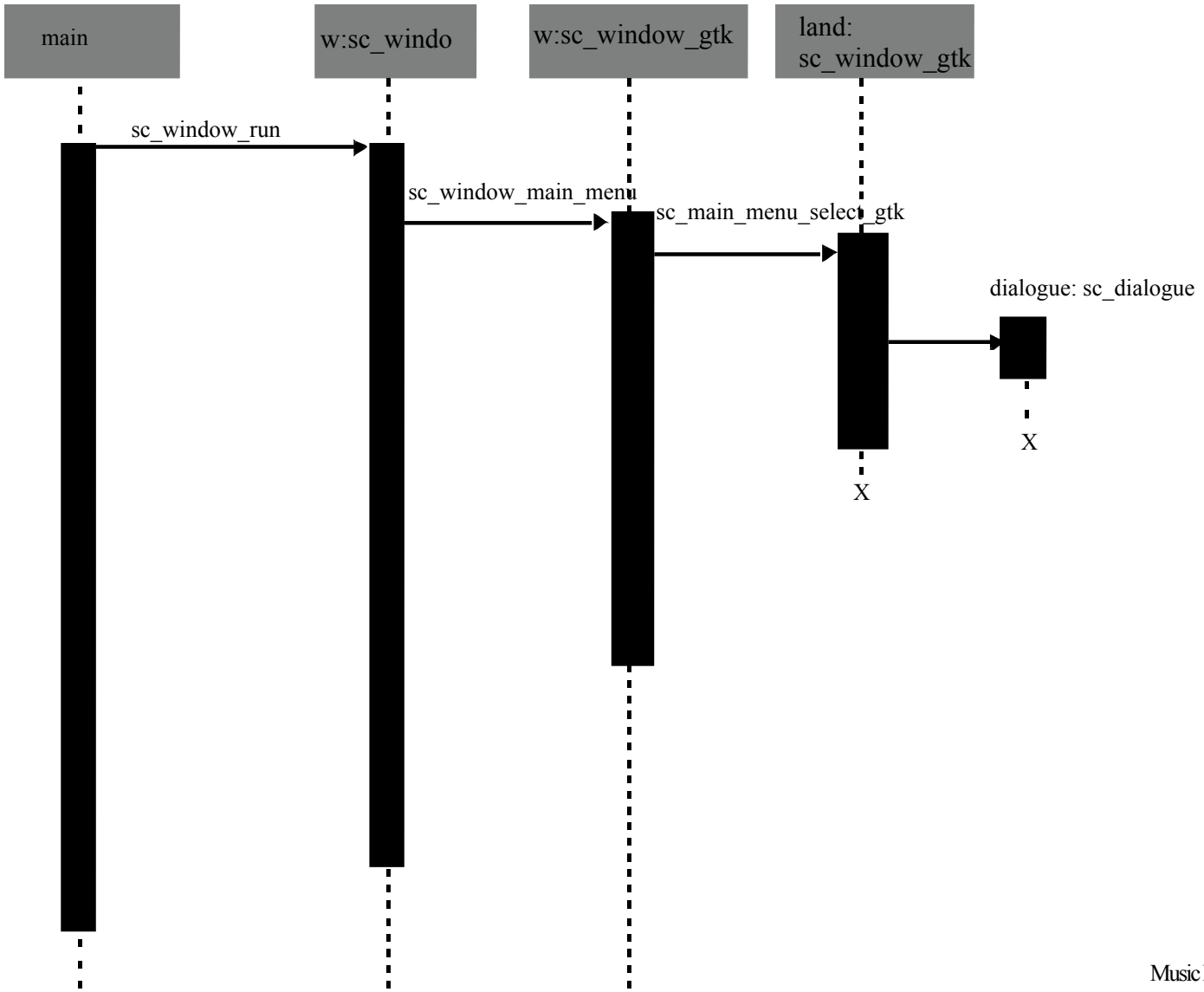
6) This diagram shows the sequence of events taken to load the landscape set up dialogue. It starts from main. There is no scenario for this feature.

7) This diagram shows the sequence of events that occurs when a player chooses to activate its teleporter by pressing 'T'. "Thinking this isn't a strategic spot to be in, she uses her teleport accessory allowing her to change position to a new random location" (Scenario 5).

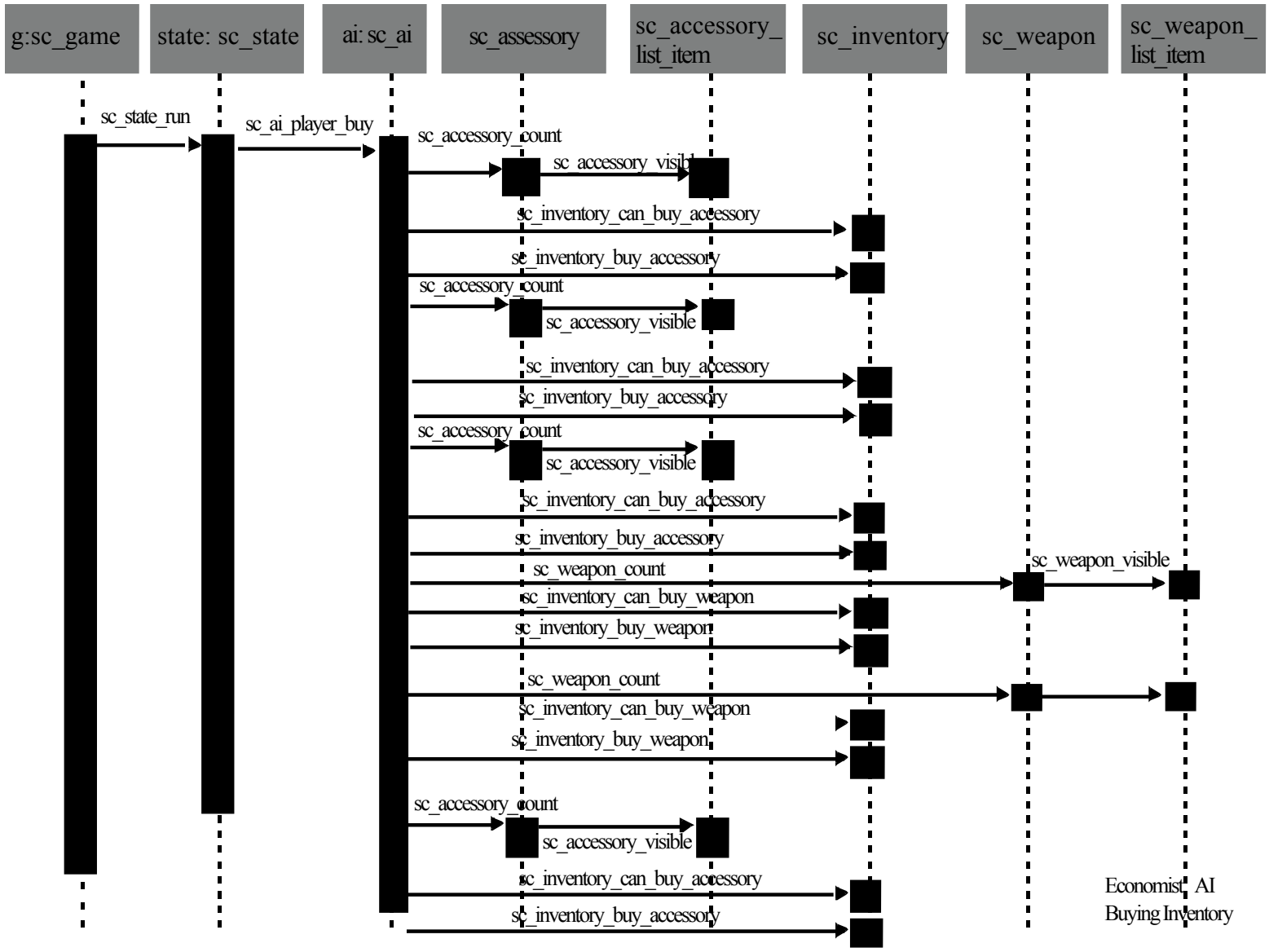
8) This diagram illustrates a sequence of events that occurs when a tank's parachute opens due to land falling from underneath it. In this scenario the tank does not slide down a hill upon hitting land again. "Luckily, she had bought a parachute in the inventory purchasing phase, and it opened up as her tank fell. This kept her tank from sustaining damage" (Scenario 5).

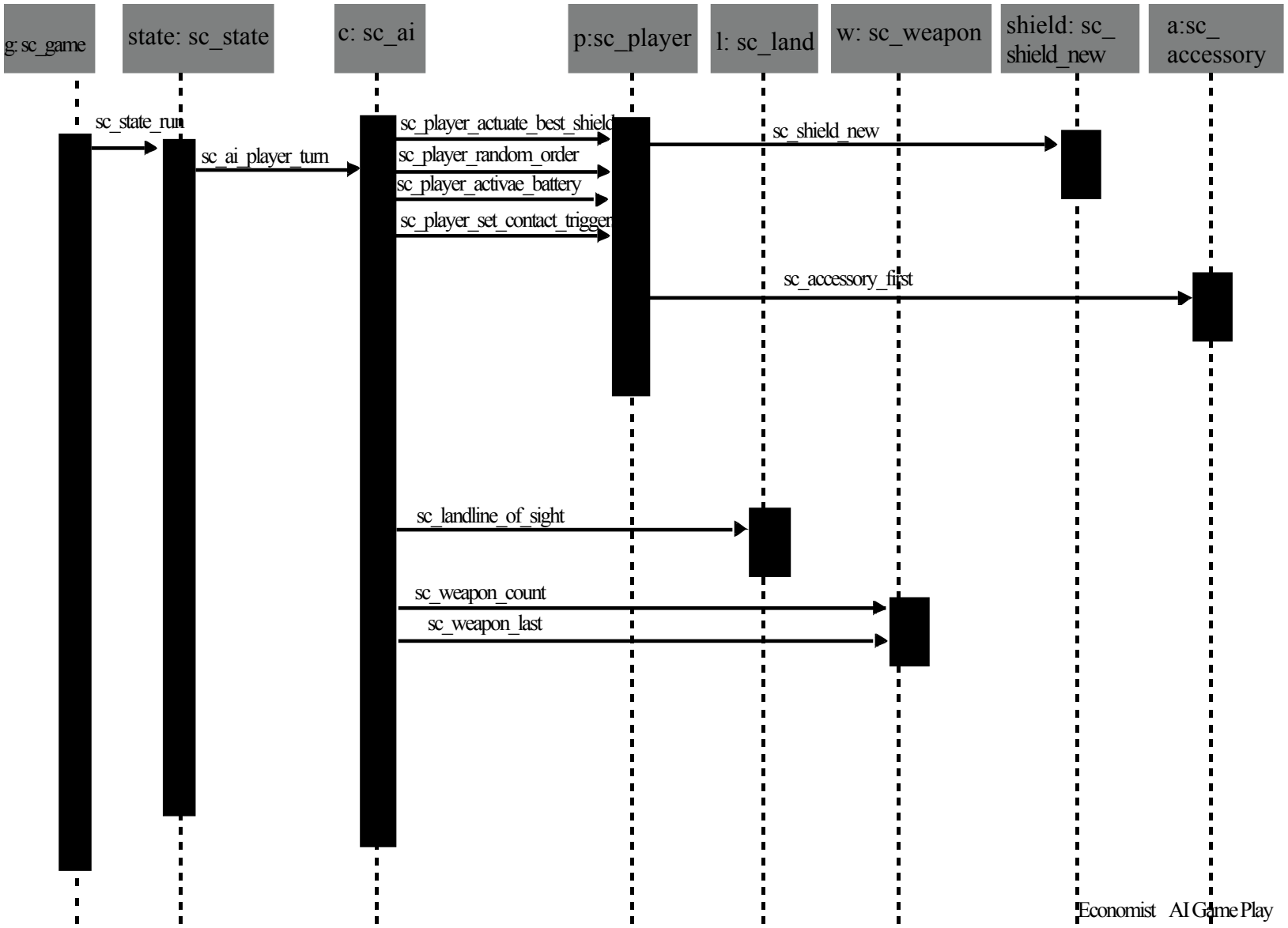
9) This diagram shows the sequence of events that occur in order to change accessory and weapon prices due to the free market economy. This occurs at the end of the round before the inventory screen pops up. "He picks the economy option and selects free market as game play.... He then pushes the start button and Xscorch moves on to the accessories/weapons menu. From here on – since he's using a free market – the prices of accessories and weapons vary in each round. The more any player buys of a certain weapon, the higher the price becomes for that weapon in each successive round. If he wishes to do so in later rounds, Bill may be able to make money by selling the weapons with inflated prices (This is a simple implementation of a supply and demand economy)" (Scenario 1).

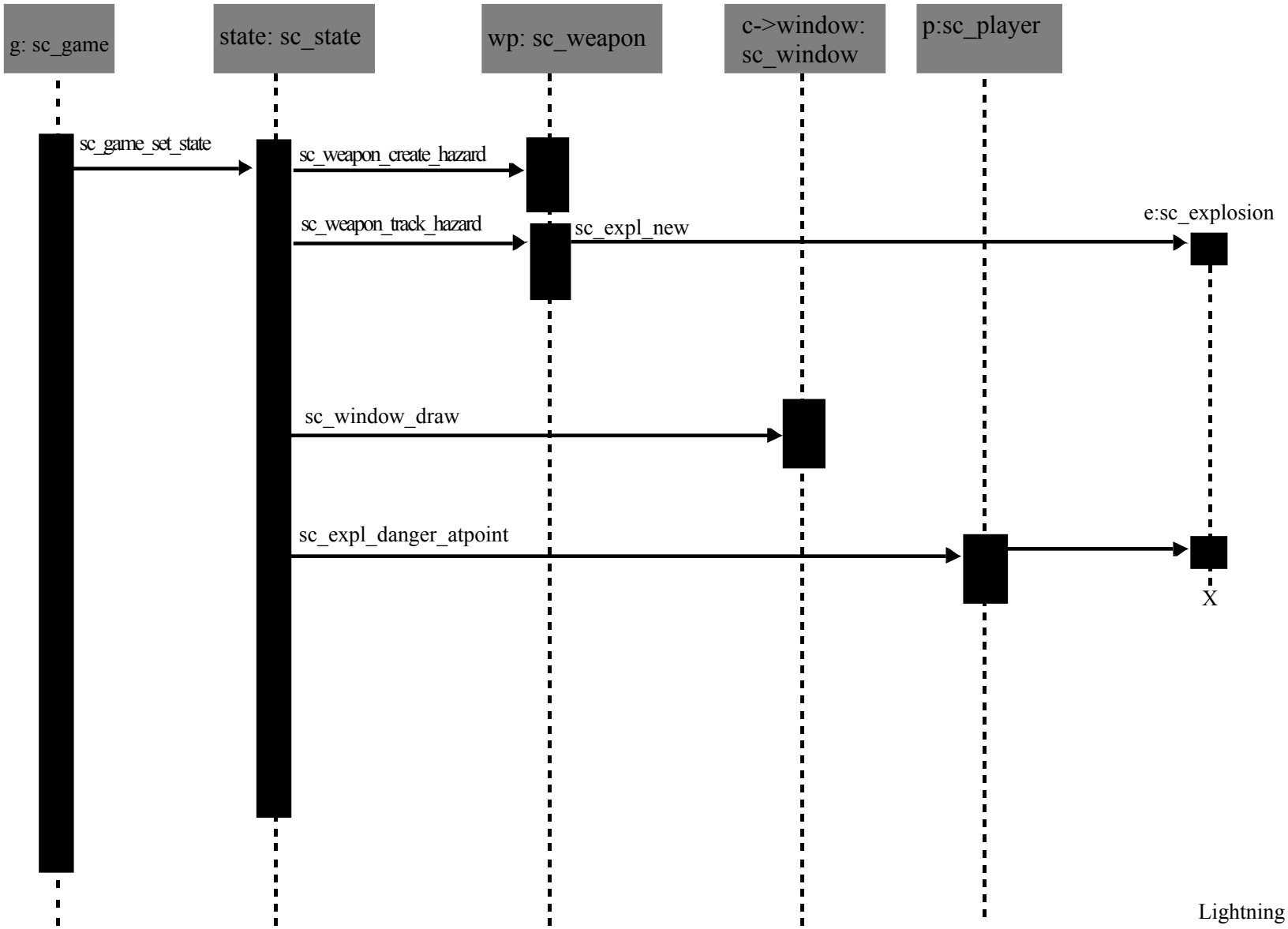
10) This diagram shows the sequence of events that occur to determine a winner in greedy scoring. This occurs at the end of the round. The scenario ends with the scoreboard being printed to the screen. "After starting Xscorch she clicks on the economics menu, where under scoring options she chooses the greedy option....She...ends up loosing her tank. But as the scoreboard comes up, Carly is pleasantly surprised to find her score to be higher then she thought it would be. That's because using greedy scoring, the person who wins isn't always the one who killed the most tanks. Instead, the final score after each round also takes into account the value of the items in the player's inventory at round's end" (Scenario 2).

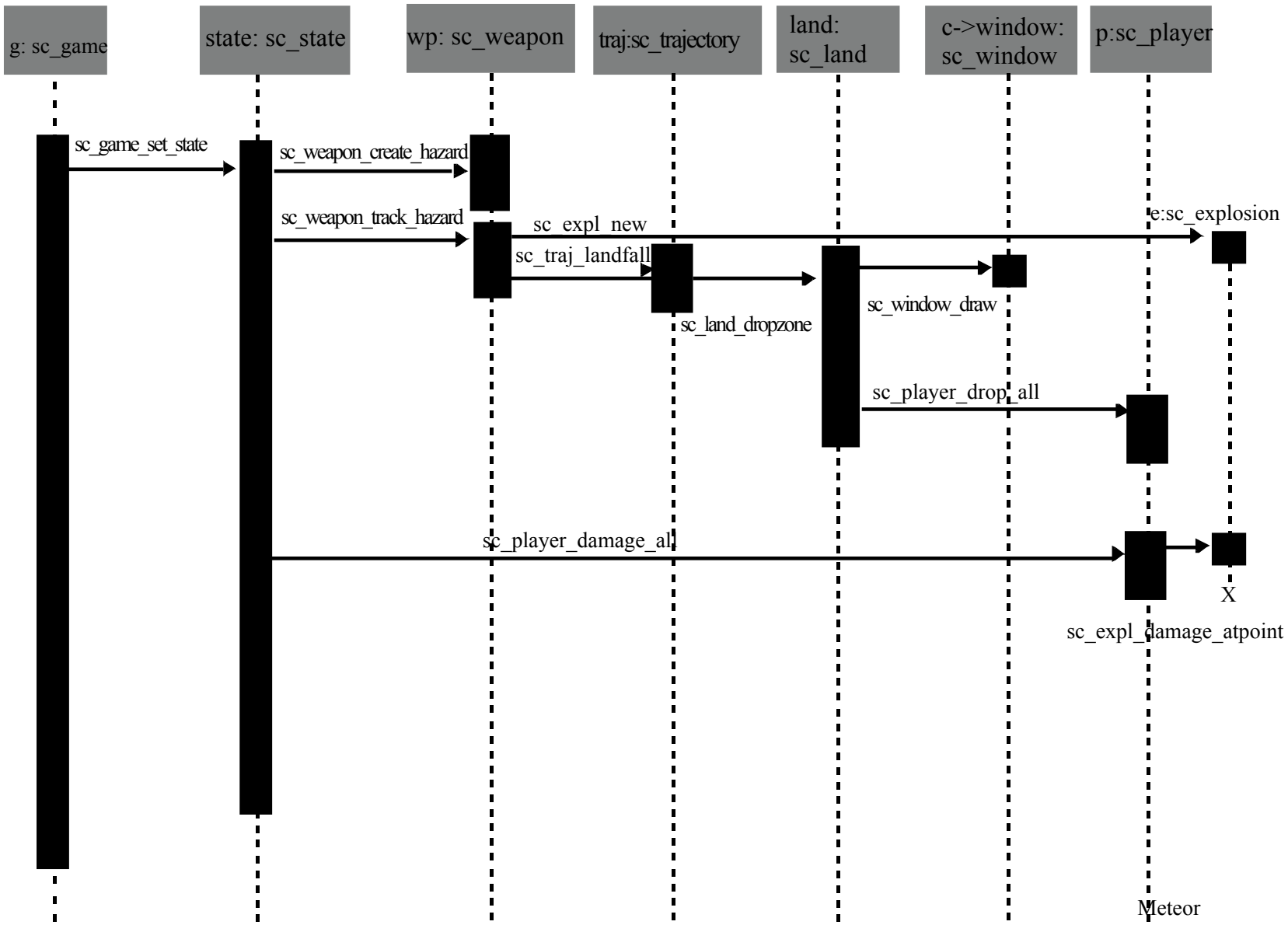


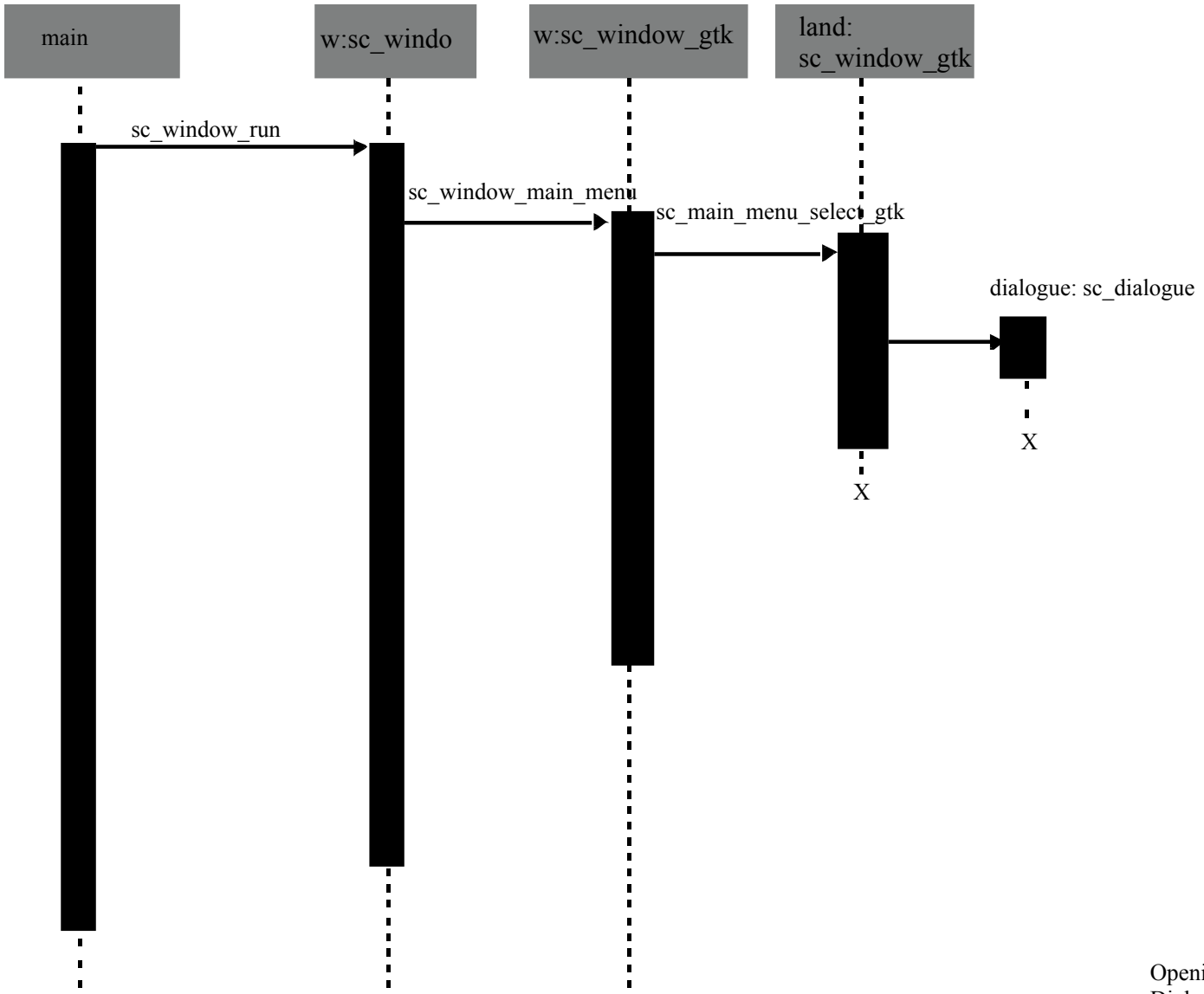
Music For Inventory


g:sc_game
state: sc_state
ai: sc_ai
sc_assessory
sc_accessory_ list_item
sc_inventory
sc_weapon
sc_weapon_ list_item
sc_state_run
sc_ai_player_buy
sc_accessory_count
sc_accessory_visible
sc_inventory_can_buy_accessory
sc_inventory_buy_accessory
sc_accessory_count
sc_accessory_visible
sc_inventory_can_buy_accessory
sc_inventory_buy_accessory
sc_accessory_count
sc_accessory_visible
sc_inventory_can_buy_accessory
sc_inventory_buy_accessory
sc_weapon_count
sc_weapon_visible
sc_inventory_can_buy_weapon
sc_inventory_buy_weapon
sc_weapon_count
sc_inventory_can_buy_weapon
sc_inventory_buy_weapon
sc_accessory_count
sc_accessory_visible
sc_inventory_can_buy_accessory
sc_inventory_buy_accessory
Economist AI
Buying Inventory

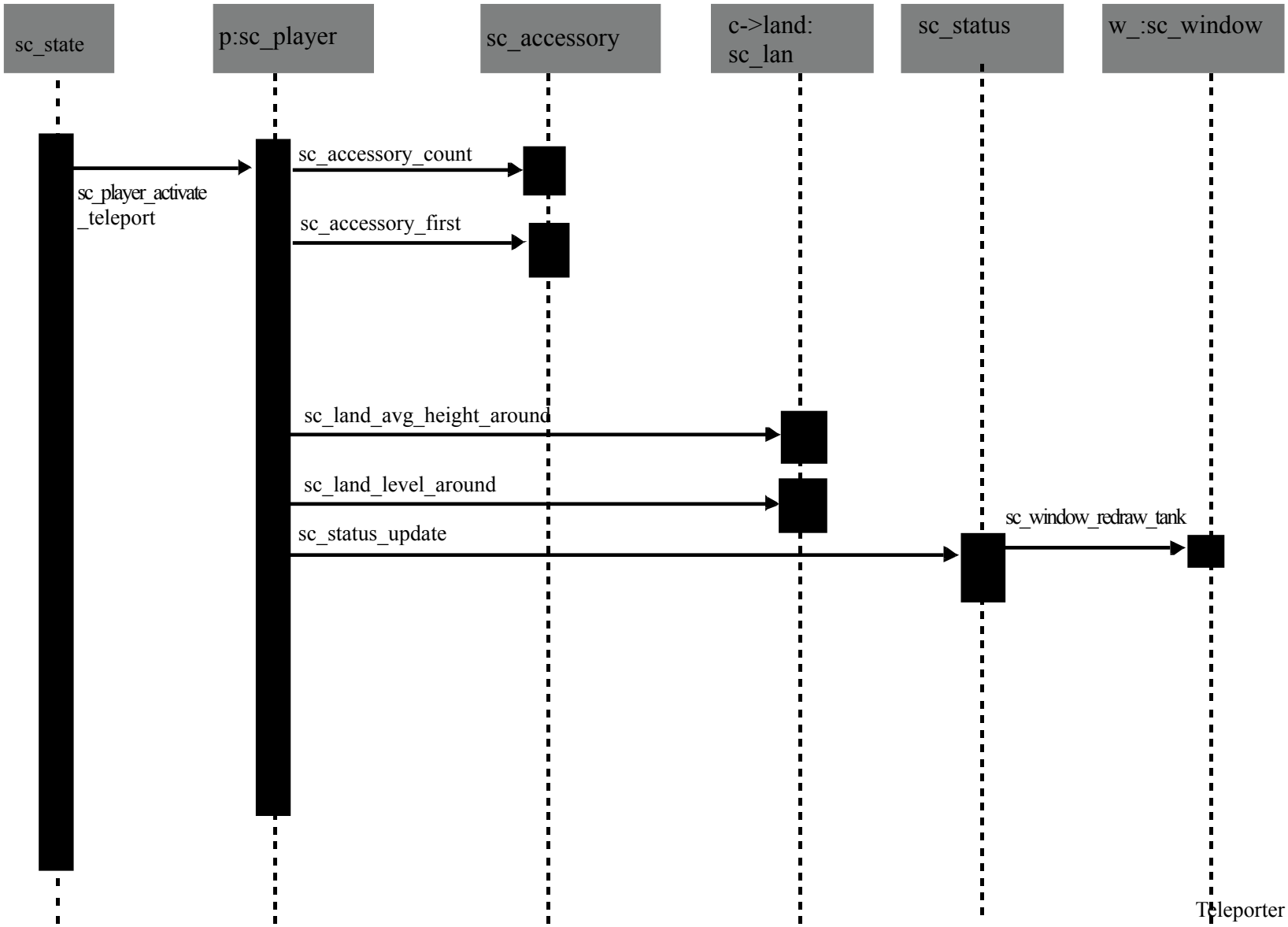

g: sc_game
state: sc_state
c: sc_ai
p:sc_player
l: sc_land
w: sc_weapon
shield: sc_ shield_new
a:sc_ accessory
sc_state_run
sc_ai_player_turn
sc_player_actuate_best_shield
sc_player_random_order
sc_player_activae_battery
sc_player_set_contact_trigger
sc_shield_new
sc_accessory_first
sc_landline_of_sight
sc_weapon_count
sc_weapon_last
Economist
AI Game Play

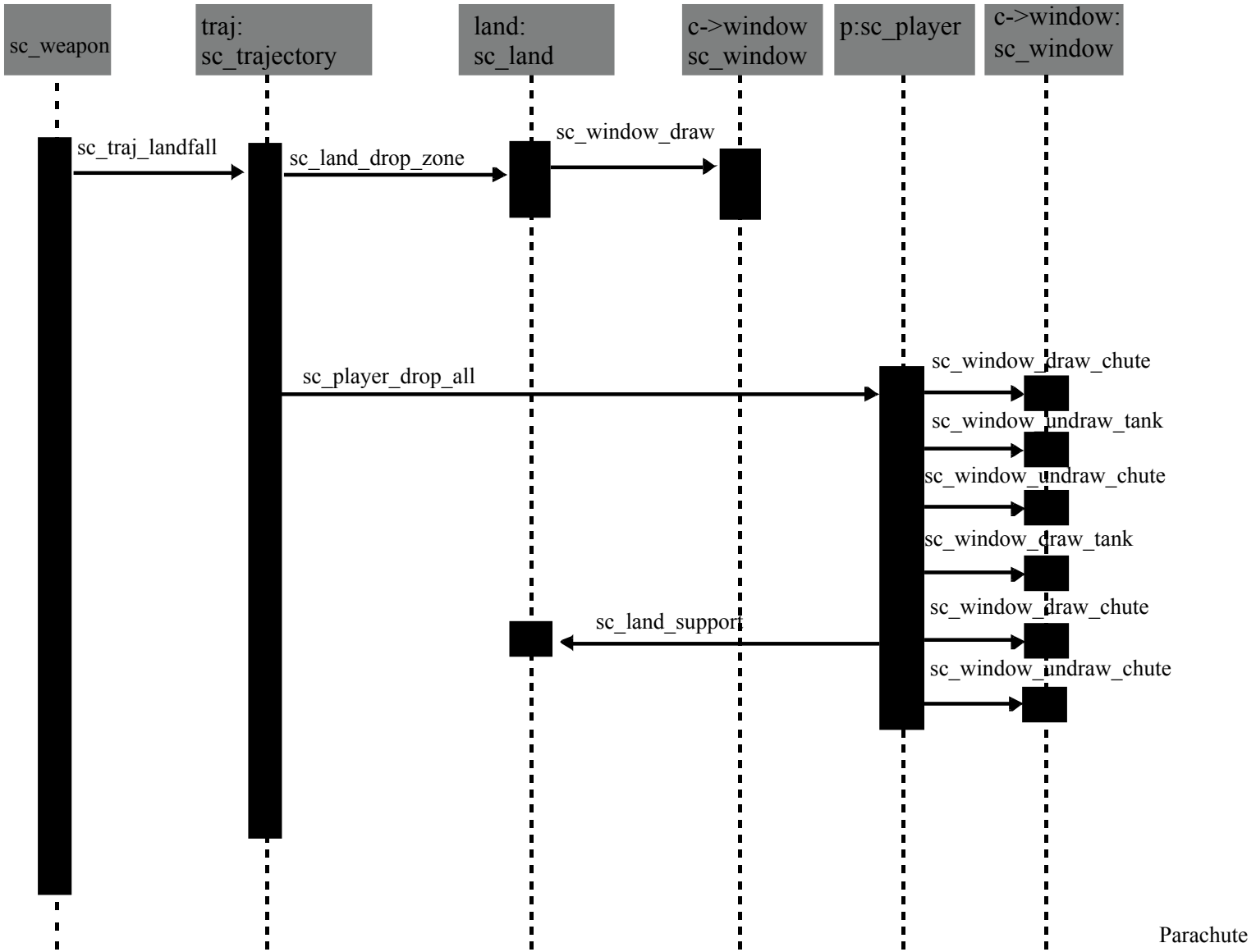

g: sc_game
state: sc_state
wp: sc_weapon
c->window: sc_window
p:sc_player
sc_game_set_state
sc_weapon_create_hazard
sc_weapon_track_hazard
sc_expl_new
e:sc_explosion
sc_window_draw
sc_expl_danger_atpoint
X
Lightning

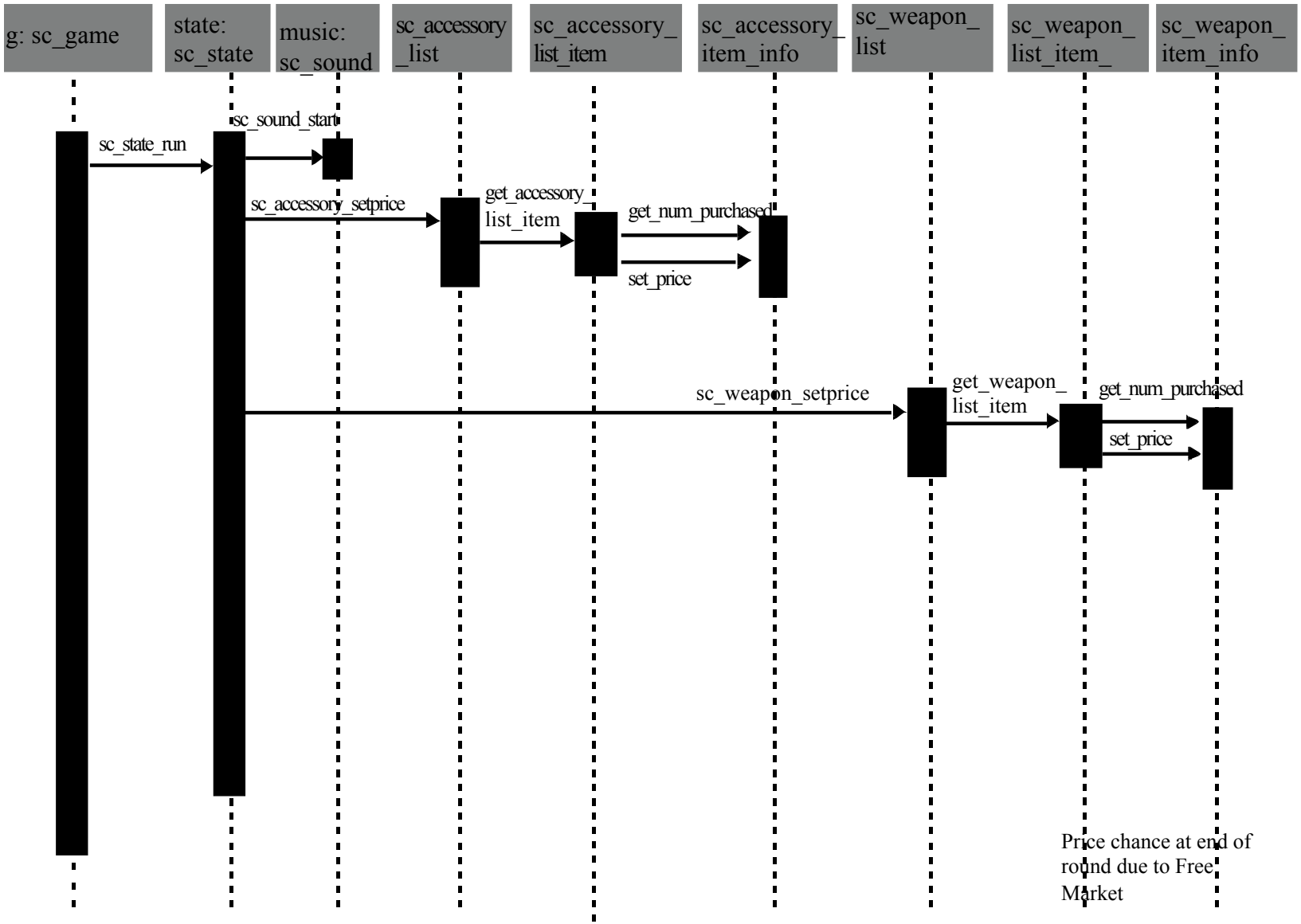

g: sc_game
state: sc_state
wp: sc_weapon
traj:sc_trajectory
land: sc_land
c->window: sc_window
p:sc_player
sc_game_set_state
sc_weapon_create_hazard
sc_weapon_track_hazard
sc_expl_new
e:sc_explosion
sc_traj_landfall
sc_window_draw
sc_land_dropzone
sc_player_drop_all
sc_player_damage_all
X
sc_expl_damage_atpoint
Meteor

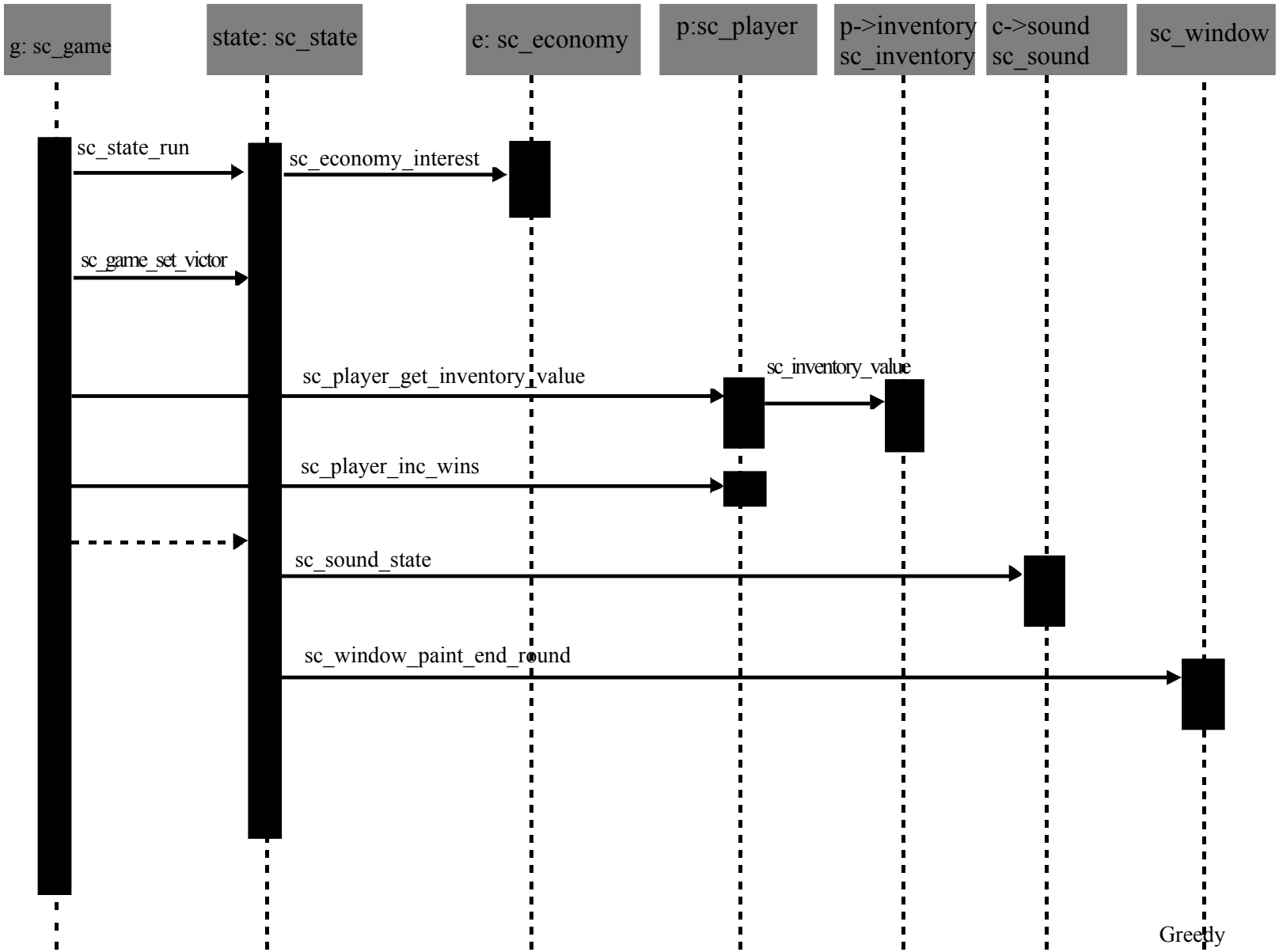


Opening landscape setup
Dialogue


sc_state
p:sc_player
sc_accessory
c->land:
sc_lan
sc_status
w_:sc_window
sc_player_activate
_teleport
sc_accessory_count
sc_accessory_first
sc_land_avg_height_around
sc_land_level_around
sc_status_update
sc_window_redraw_tank
Teleporter


sc_weapon
traj:
sc_trajectory
land:
sc_land
c->window
sc_window
p:sc_player
c->window:
sc_window
sc_traj_landfall
sc_land_drop_zone
sc_window_draw
sc_player_drop_all
sc_window_draw_chute
sc_window_undraw_tank
sc_window_undraw_chute
sc_window_draw_tank
sc_window_draw_chute
sc_land_support
sc_window_undraw_chute
Parachute


g: sc_game
state: sc_state
music: sc_sound
sc_accessory _list
sc_accessory_ list_item
sc_accessory_ item_info
sc_weapon_ list
sc_weapon_ list_item_
sc_weapon_ item_info
sc_sound_start
sc_state_run
sc_accessory_setprice
get_accessory_ list_item
get_num_purchased
set_price
sc_weapon_setprice
get_weapon_ list_item
get_num_purchased
set_price
Price chance at end of round due to Free Market


g: sc_game
state: sc_state
e: sc_economy
p:sc_player
p->inventory
sc_inventory
c->sound
sc_sound
sc_window
sc_state_run
sc_economy_interest
sc_game_set_victor
sc_player_get_inventory_value
sc_inventory_value
sc_player_inc_wins
sc_sound_state
sc_window_paint_end_round
Greedy

Open Issues

The exact implementation of team play has not been worked out yet. We plan on having another variable in the sc_player specifying what team a player is on, and modifying AI aiming functions so that they do not target their own team members. Also, two of our features do not easily lend themselves to implementation in synchronous battle mode. The first is hostile environment. We have two ideas on how to do this: 1) only one meteor/lightning strike per set of turns or 2) one possible strike per turn before anything is resolved. The other feature is teleporters. There are three options for this at this time: 1) instantly move the tank so the player can then aim accordingly, 2) when everything gets resolved, fire the gun, then move the tank, or 3) when everything gets resolved, move the tank, then fire the gun.