### Функциональные требования

1. Пользователь открывает программу
2. Открывается меню
   1. Новая игра
   2. Продолжить
   3. Таблица рекордов
   4. Выйти
3. Если новая игра, то:
   1. Создаётся новая карта
4. Если продолжить, то:
   1. Загрузить сохранение
5. Карта отрисовывается
6. Пользователю предлагается набор действий
   1. Передвижение
   2. Открытие инвентаря
   3. Открытие меню
      1. Сохранение + выход
      2. Посмотреть управление
      3. Закрыть меню
7. Если пользователь выбрал передвижение:
   1. Клетка свободна – идёт
   2. На клетке монстр – атака
   3. На клетке стена – не идём
   4. На клетке предмет – подбираем
8. Если пользователь открыл инвентарь:
   1. Отрисовка инвентаря
      1. Предлагаем пользователю выбрать предмет
         1. Если экипировка – экипировать
         2. Если использование – использовать
         3. Вернуться в инвентарь
      2. Выход из инвентаря
9. Если пользователь открыл меню – пункт 2
10. Если пользователь встал на клетку лифта – переход на след. уровень
11. Если пользователь взял – конец игры:
    1. Вывести поздравления
    2. Сохранить результат в таблицу лидеров
    3. Вывести таблицу лидеров
12. Если здоровье игрока <= 0:
    1. Вывести экран поражения
    2. Сохранить результат в таблицу лидеров
    3. Вывести таблицу лидеров
13. Если на пользователе эффекты:
    1. Применить эффект
    2. Срок эффекта уменьшается

### Структурная схема

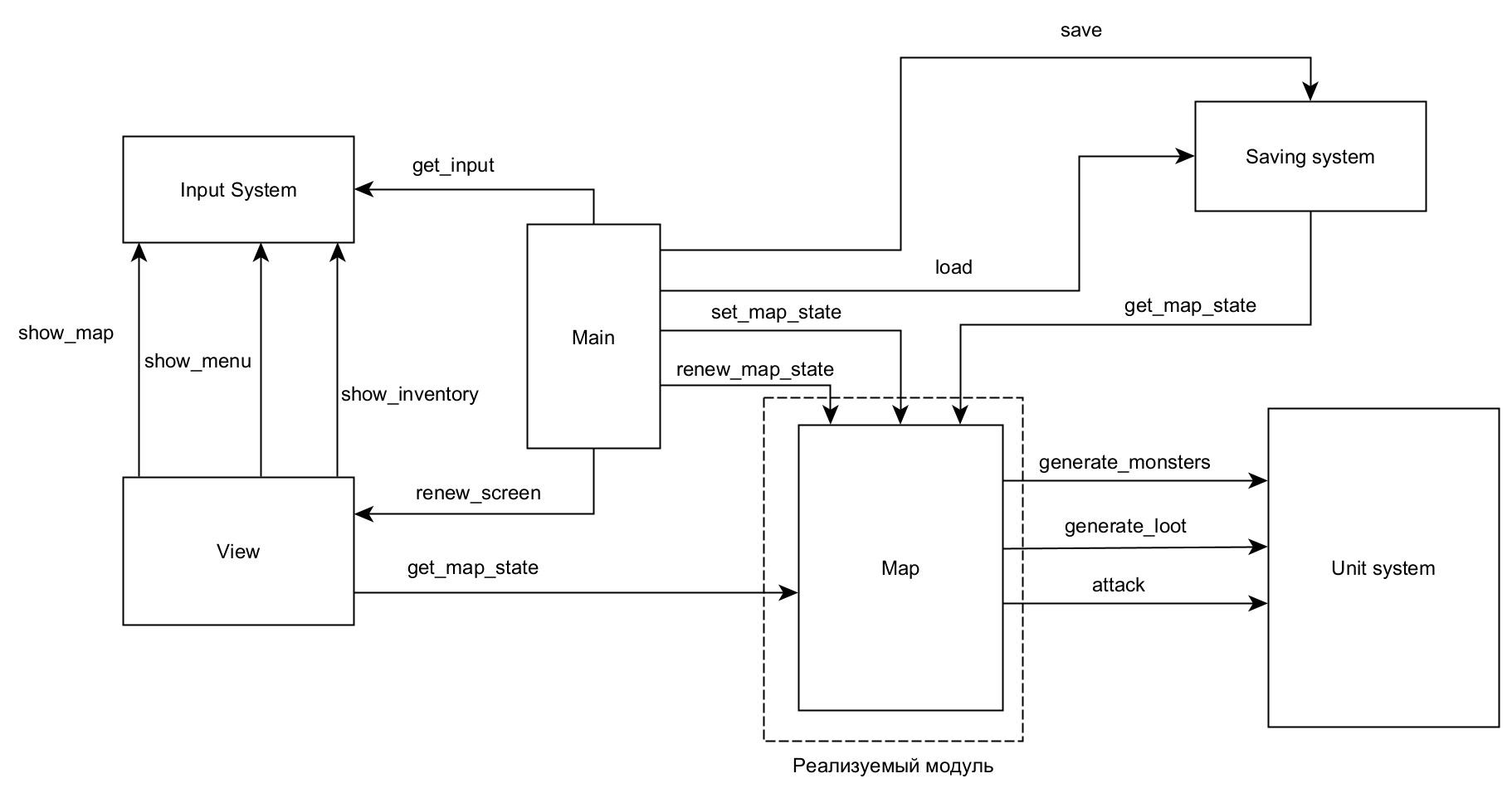


рис. 1. «Структурная схема»

### Прототипы функций

/\*\*  
\* @brief генерирует ландашфт игровой карты (на основе алгоритма игры "жизнь")  
\* @param game\_map указатель на карту, ландшафт которой будем генерировать  
\* @return код ошибки  
\*/  
int generate\_maps\_landscape(GameMap \*game\_map);

/\*\*

\* @brief размещает монстров и предметы на игровой карте

\* @param game\_map игровая карта (с готовым ландшафтом)

\* @param items массив предметов

\* @param items\_num кол-во предметов

\* @param units массив мобов

\* @param units\_num кол-во мобов

\* @return код ошибки

\*/

int place\_objects\_on\_map(GameMap \*game\_map, Item \*items, int items\_num, Unit \*units, int units\_num);

/\*\*

\* @brief обновляет состояние игровой карты

\* @param game\_map игровая карта

\* @return код ошибки

\*/

int renew\_map\_state(GameMap \*game\_map);

/\*\*

\* @brief проверяет, окончилась ли игра на игровой карте

\* @param game\_map игровая карта

\* @param is\_finished сюда вернётся 1, если игра закончилась, и 0 в противном случае

\* @return код ошибки

\*/

int game\_is\_finished(GameMap \*game\_map, char \*is\_finished);

/\*\*

\* @brief возвращает указатель на игровую карту

\* @param game\_map игровая карта

\* @return код ошибки

\*/

GameMap\* get\_map\_state(GameMap \*game\_map);

/\*\*

\* @brief копирует одну игровую карту в другую

\* @param game\_map игровая карта -- в которую будем копировать

\* @param source откуда будем копировать

\* @return код ошибки

\*/

int set\_map\_state(GameMap \*game\_map, GameMap \*source);

/\*\*

\* @brief создаёт карту, выделяет память и задаёт некоторые стартовые параметры (см. в settings)

\* @param game\_map указатель на игровую карту, которую надо инициализировать

\* @param settings параметры создаваемой карты

\* @return код ошибки

\*/

int init\_map(GameMap \*game\_map, MapSettings settings);

/\*\*

\* @brief чистит память, которую занимает карта

\* @param game\_map указатель на игровую карту, за которой надо почистить память

\* @return код ошибки

\*/

int delete\_map(GameMap \*game\_map);

### Прототипы функций из других модулей

### Asdsad genenate\_monsters

### generate\_loot

### attack