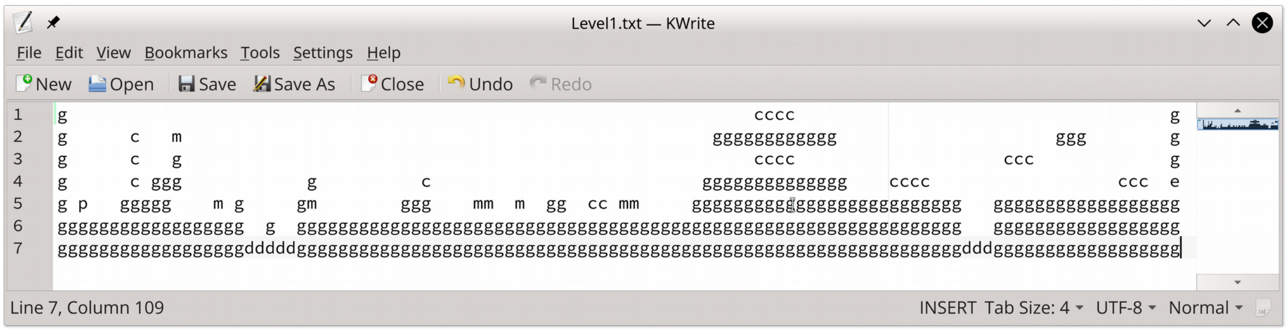
1. Introduction

This guide describes how to add new gameplay features to the project. Mainly it means adding new units to game, adding new levels and change behavior of the existing units.

2. How to add new level

To add new level you should create new level's map and add it to the game initialization.

Our project uses MapParser class for generating levels from text files. Text file of level must consists of characters, where each character means one unit in this position in the level. For example there is content of the Level1.txt file which is the first level of our game (Picture 2.1):



```
g                                                                  cccc                                    g
g      c   m                                                   gggggggggggg                     ggg        g
g      c   g                                                       cccc                    ccc             g
g      c ggg            g          c                          gggggggggggggg    cccc                  ccc  e
g p   ggggg    m g     gm        ggg    mm  m  gg  cc mm     gggggggggggggggggggggggggg   gggggggggggggggggg
gggggggggggggggggg  g  gggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg   gggggggggggggggggg
gggggggggggggggggggdddddggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggdddgggggggggggggggggg
```

Picture 2.1 — example of text representation of level

Game searches levels in the "Levels" subdirectory in the directory wich contains game executable or, if it doesn't exists, in the "../../Levels" directory (this option is only for running project from build tree). In build tree put your levels to the "Mario/Levels" directory.

Meaning of different characters you will find in the Table 2.1:

Table 2.1 – meaning of characters for MapParser

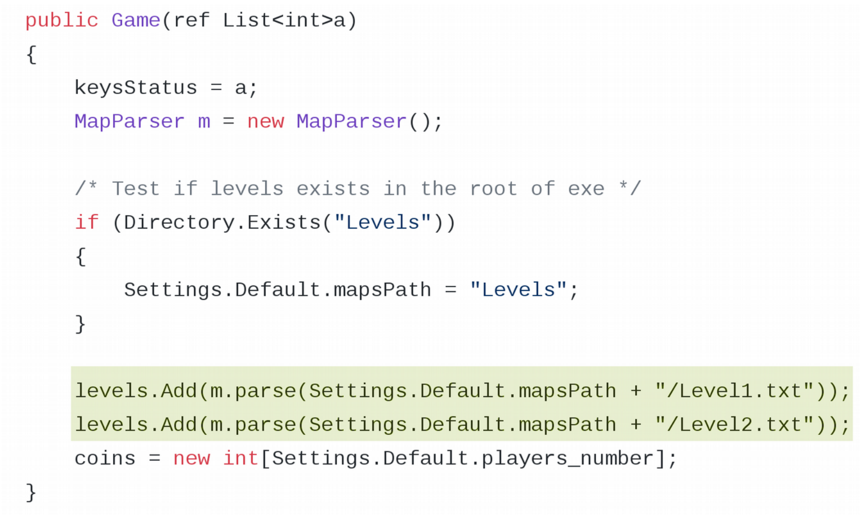
Character	**Object of class which it is parsed to**	**Short description of unit**
g	GroundUnit	Base ground unit
p	Player	Player
m	Mushroom	Mob which goes on ground

Character	**Object of class which it is parsed to**	**Short description of unit**
d	DieUnitBlock	Transparent unit which is used for killing of everithing which has contact with it. Usually it is on the bottom of abysses
e	Door	Unit which cause completion of level
c	Coin	Coin
b	Bird	Flying mob
G	GroundPlatform	Flying horizontally GroundUnit

Pay attention: do not use tabulators in your maps, because tabulation is one character which is displayed as some more spaces.

Other not defined characters means space (no unit in this place).

After your level's map was created, you should add it to the Game constructor like existing levels (Picture 2.2):



```
public Game(ref List<int>a)
{
    keysStatus = a;
    MapParser m = new MapParser();

    /* Test if levels exists in the root of exe */
    if (Directory.Exists("Levels"))
    {
        Settings.Default.mapsPath = "Levels";
    }

    levels.Add(m.parse(Settings.Default.mapsPath + "/Level1.txt"));
    levels.Add(m.parse(Settings.Default.mapsPath + "/Level2.txt"));
    coins = new int[Settings.Default.players_number];
}
```

Picture 2.2 – adding levels to game

Note that levels will appear in game in order they was added to the game in constructor.

3. How to add new unit

To add new unit to game you should do three steps:

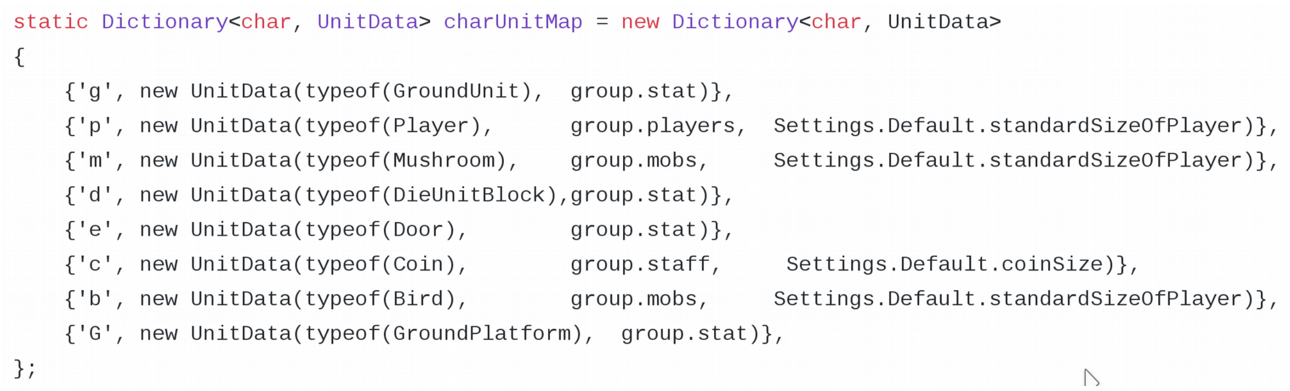
1. create new class of this unit
2. describe its interaction with other units if needs

3. add it to map parser

Your class should be inherited from Unit class or some other existing class inherited from it. If you create new mob, inherit you class from Mob, if you create some ground or static unit inherit in from GroundUnit, e.t.c. (see unit_hierarchy UML diagram).

If your unit should interact with other units in some specific way, you can describe this behavior in World class in the set of functions describing collisions: *sideIndependentAction*, *horizontalSpecificAction*, *bumpToLeft*, *bumpToRight*, *bumpToUp*, *bumpToDown.*

To add your unit to MapParser you should add it to the *charUnitMap* like other units are (Picture 3.1):



```
static Dictionary<char, UnitData> charUnitMap = new Dictionary<char, UnitData>
{
    {'g', new UnitData(typeof(GroundUnit),  group.stat)},
    {'p', new UnitData(typeof(Player),      group.players,  Settings.Default.standardSizeOfPlayer)},
    {'m', new UnitData(typeof(Mushroom),    group.mobs,     Settings.Default.standardSizeOfPlayer)},
    {'d', new UnitData(typeof(DieUnitBlock),group.stat)},
    {'e', new UnitData(typeof(Door),        group.stat)},
    {'c', new UnitData(typeof(Coin),        group.staff,     Settings.Default.coinSize)},
    {'b', new UnitData(typeof(Bird),        group.mobs,     Settings.Default.standardSizeOfPlayer)},
    {'G', new UnitData(typeof(GroundPlatform),  group.stat)},
};
```

Picture 3.1 — units in MapParser

The first field is the character wich will represent your unit in the text files with maps, in UnitData the first parameter is your unit type, second is group of your unit (see introduction: collision matching), third parameter (optionally) is the size of side of your unit. Size is standardSizeOfUnit by default.