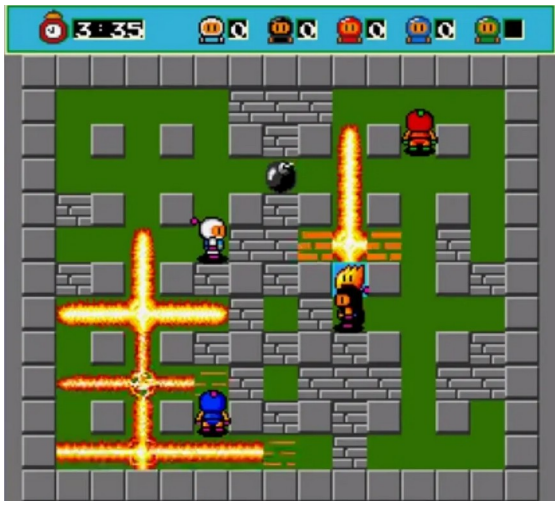
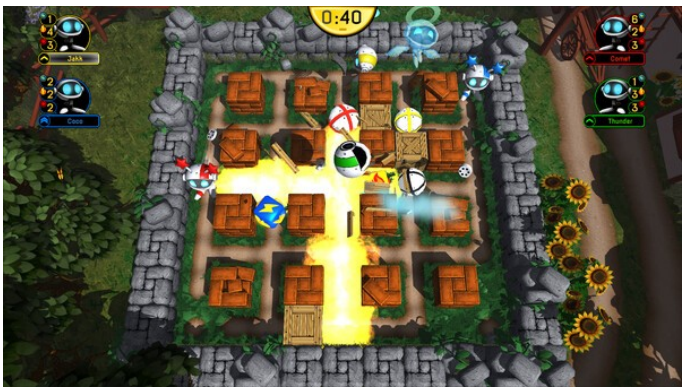
# Multiplayer Bomberman

During the semester, a competitive multiplayer Bomberman game needs to be implemented. For similar games see the classic Bomberman game (better known in Europe as Dyna Blaster), their modern 3D versions, or any game similar to these.

You can play the original here: https://www.retrogames.cz/play_455-DOS.php



*Dyna Blaster (1990)*



*Dyna Blaster (2023)*

## Brief description of the game

The game is played on a 2-dimensional field, consisting of square-shaped tiles. The game is played by 2 players, each controlling a Bomberman character with the goal of being the last one standing. The game field contains wall elements, boxes, monsters, and the players' characters. Players can place bombs to blow up boxes, monsters, and players (including themselves). A player loses (and thus their opponent wins) if they explode or are caught by a monster.

### *Walls*

The wall elements are indestructible, and with a few exceptions detailed in the optional subtasks, neither the players nor the monsters can step on them. The game field is surrounded by a wall (so you cannot step out of it).

### *Players*

The players each control a Bomberman character, which can move left, right, up, and down. As a special action, players can place a bomb on the tile they are standing on. A player can only have one bomb placed at a time (see the Boxes section for more on this). Players can control their character using the keyboard.

### *Bombs*

The bombs explode after a short time, affecting not only the current tile but also 2 tiles in all 4 directions (left, right, up, down). The walls are not destructible and the explosion does not spread through them. The boxes can be blown up, but they absorb the force of the explosion, preventing it from spreading through them. The bombs can also detonate each other before their timers run out, creating a chain reaction. The explosion spreads quickly, but not instantly, so the effect of an explosion on further tiles is delayed proportionally to the distance from the bomb's explosion position. The exploding bomb destroys players and monsters on the affected tiles.

With a few exceptions detailed in the optional subtasks, neither the players nor the monsters can step on the bombs. (Players place the bomb "under" themselves, from which they can still step off, but they cannot step back onto that tile while the bomb is there.)

### *Boxes*

The boxes can be blown up, and with a few exceptions detailed in the optional subtasks, neither the players nor the monsters can step on them. If a box is blown up, a power-up may appear from it, which, if any player steps on it, gives a bonus to their character for that game. The possible bonuses are:

- Number of bombs: The number of bombs that can be placed by the player increases by 1.
- Blast range: The blast range of the player's bombs increases by 1 tile in all 4 directions.

### *Monsters*

Monsters also roam the game field, and if they step onto the same tile as any player's character, the player dies. In the basic task, the monsters should have a simple heuristic: when they hit an obstacle, they randomly change direction. Occasionally, they should also change their direction of movement by their own volition to make their movement less predictable.

### *End of the game*

If a player explodes or is caught by a monster, the game should continue for a short time (for example, for the duration of the bomb timers). If the other player can stay alive during this time, they have won the game. If the other player also dies during this time, the game ends in a draw.

### *Starting the game*

When starting a new game, we should be able to choose from at least 3 pre-made, different maps. On each map, the positions of the walls and boxes can be predefined, but which boxes hide a bonus (and what kind) should be determined dynamically at runtime. The base area of the maps can be rectangular, but other shapes are also possible.

We should be able to specify how many times a player has to win to win the map. After each game, the current game result should be displayed, as well as how many games each player has won. (A draw counts as a loss for both players.) After the last game, the final result should also be displayed.

## Subtasks

The complexity of the basic game is **2 units**. (1.5 units for teams of 4, 3 units for teams of 2). Each team needs to select from the following subtasks during the task specification so that the **project complexity reaches 5 units**. Other subtasks can also be implemented based on your own ideas, after consultation with the instructor.

The complexity value of the functionality actually implemented by the end of the semester serves as the upper limit of the grade for each member of the team.

### *Intelligent Monsters [1 complexity]*

We should encounter different types of monsters on the game fields, which move at different speeds and according to different heuristics. The individual monsters should also appear visually different. At least 4 different monsters need to be implemented:

1. The monster featured in the basic task.
2. A monster similar to the one in the basic task, which can go through walls and boxes. (But not bombs.) If it reaches an obstacle, i.e., a wall or a box, and there is still a non-obstacle square on the field straight ahead (after the obstacles), it should not change direction, but continue in its direction of movement. In this case, if there are several obstacle fields in a row, it should go through them without changing direction. It should move slower than the basic monster.
3. A monster similar to the one in the basic task, but when it hits an obstacle and changes direction, it should choose a direction that starts towards the player closest to it based on the shortest path. (If it cannot reach any of the players due to obstacles on the field, it should wander randomly.) It should move faster than the basic monster.
4. A monster similar to the previous one, but it should be able to change direction also at a fork in the road. With a certain probability, it should make a wrong decision and choose the wrong path. It should move at the same speed as the basic monster.

### *Advanced Power-ups [1 complexity]*

In addition to the 2 power-ups defined in the basic task, more complex bonuses should also be available at the location of the blown-up boxes:

- Detonator: The bombs placed by the player should not explode due to a timer. After placing the last bomb, all the player's bombs should explode when using the bomb placement function one more time.
- Roller skate: The speed of the player's character should increase. It is not cumulative, the pickup of any second and further roller skates have no effect.
- Invincibility: For a short time, the player's character should be invincible. Visually indicate that the player is currently invincible, and also when its effect is about to end.
- Ghost: For a short time, the player's character should be able to pass through walls, boxes, and bombs. Visually indicate that the player has this ability and also when its effect is about to end. If the player is on a box or wall tile at the end of the ability, they die. (They can step off a bomb.)
- Obstacle: The player should be able to place obstacles that behave like boxes in terms of

game logic, but no bonus (power-up) ever comes out of these. A maximum of 3 obstacles can be placed by a player. (If they are blown up, a new one can be placed.) This ability should be cumulative, i.e., with the possible second pickup of the bonus, 6 obstacles can be placed, etc.

### *Hindering Curses [0.5 complexity]*

"Curses" with disadvantages can also appear from the boxes. In this case, the specific disadvantage should not be known in advance for the players, but the same graphic should indicate all of them (e.g., a skull in the original game). The user can empirically discover the actual disadvantage after picking it up. Possible disadvantages:

- The speed of the player's character should decrease for a while.
- The blast range of the bombs placed by the player should be only 1 field for a certain time.
- The player should not be able to place bombs for a while.
- The player's character should immediately place the bombs as soon as they can (they are on a free field and can place a bomb). This disadvantage should also only have an effect for a certain time.

### *Three Players [1 complexity]*

The game should not only be playable by 2, but also by 3 players. When starting a new game, the number of players should be selectable. The game lasts until only one player remains alive in the case of 3 players. (The game result is still a draw if they also die within a short time.)

### *Level Editor [1 complexity]*

The application should include a full-fledged level editor feature, where it is possible to define wall elements and boxes, as well as the starting positions of the players. For monsters, either their starting position or their type and number should be definable - in the latter case, the monsters are dynamically placed on the field when the game starts. The designed levels should be saveble within the program, editable later, and of course, playable. The designed levels should also be exportable to a file and importable on another computer.

### *Persistence [0.5 complexity]*

There should be an option to save a given game state, including the number of games each player has won so far. Later, it should be possible to reload a selected game state and continue the game. It should also be possible to manage multiple saves.

### *Customizable Controls [0.5 complexity]*

There should be an option to customize the key layout assigned to player control. This setting should also be saved, and the last saved setting should be in effect when the application is restarted.

### *Continuous Movement [0.5 complexity]*

The movement of players and monsters between the fields of the game board should not be jumping, but continuous. Characters can still logically be exclusively on a single field, but their

movement between fields should require multiple presses (or holding down) of the control buttons.

### *2.5D Graphics [0.5 complexity]*

The basic task requires an overhead graphics implementation, where each object appears within its own field. This subtask requires to implement 2.5-dimensional graphics where objects visually extend beyond their own cell (e.g., a wall covers the foot of the player character on the field above it).

### *3D Graphics [1 complexity]*

The basic task requires an overhead graphics implementation, where each object appears within its own field. This subtask is to implement a real, rotatable 3-dimensional display.

### *Networking [1 complexity]*

The program should be supplemented with networking functionality, and thus the game should be playable on two computers. If this task is chosen, the implementation of the local multiplayer function is no longer mandatory, it can be completely replaced by the multi-machine mode.

### *Battle Royale [0.5 complexity]*

At the start of each game, a countdown should start, clearly visible to the players. When the countdown ends, the game area should shrink, e.g., by making the fields on the edge of the map unusable. After this, a new countdown should start, but with a shorter time frame. If a monster or a player is on a field that is thus removed from the game area, they die.